SÓLO PROGRAMADORES

Revista especializada para usuarios de PC

AÑO 2 Nº 8
1250 PTAS.

ESCRITA por desarrolladores para desarrolladores

Entrevista a Digital Dreams Multimedia

Arquitectura RISC

Cómo es una red neuronal

Análisis de C Set ++ para OS/2

Y además:

- Formatos gráficos II
- Acceso a Internet
- Instalación de LINUX
- Aprender a programar
- Curso de C++
- Red local LANtastic

DE REGALO
Archivador, fichas de programación y un disco









CARTA DEL DIRECTOR

UNO DE LOS NUESTROS

Quince de marzo, corrían las 11 de la noche en mi reloj y estábamos *a punto* de cerrar el presente número de SÓLO PROGRAMADORES. En la redacción quedábamos *los de siempre*. De pronto, sonó el teléfono: "Hola buenas, ¿es la redacción de la revista? Llamaba por el tema del concurso, sí, ya sé que hoy acababa el plazo pero el caso es que al final me *lié* optimizando unas rutinas... Si os lo llevo mañana, ¿podríais admitirlo?" Aunque no solemos hacer excepciones, dijimos que sí. Era difícil negarse. Las once y pico de la noche y a uno de nuestros lectores se le había ido *el santo al cielo* por optimizar una rutina para participar en el concurso de las Cuatro en Raya. Se trataba sin duda de *uno de los nuestros*, adictos a esa especie de fiebre llamada programación cuyo infalible tratamiento es sentarse unas horas *a hacer algo* delante del ordenador. Seguro que las más de 100 personas que han participado en este primer torneo han necesitado días o incluso semanas hasta llegar a su solución personal del problema y seguro también que, durante todo ese tiempo, han disfrutado poniendo a prueba su capacidad para afrontar y resolver problemas. Ahora ya sólo falta saber cuál es el mejor, quizá en el próximo número...

Este mes, como ya habrán observado todos los lectores comenzamos a entregar con la revista una serie de fichas coleccionables sobre aplicaciones y programación. El objetivo es tratar de sintetizar el mayor número de información en un formato muy manejable, para tenerlas al lado del ordenador y utilizarlas como referencia en cualquier momento. De esta forma, en lugar de tener que ir al manual de 300 ó 400 páginas para averiguar cuál era la tecla para compilar sin tener que ir al menú, o la forma más rápida de cambiar el modo gráfico en una aplicación, sólo habrá que buscar la ficha correspondiente al lenguaje o aplicación para encontrar la información más importante sintetizada. Mes a mes iremos entregando nuevas fichas con información de programas, lenguajes, sistemas operativos, aplicaciones shareware, etc. En definitiva, toda la información imprescindible que un programador debe tener al lado de su ordenador para perder el menos tiempo posible al trabajar. Esperamos que esta iniciativa sea del agrado de la mayoría de los lectores y que, incluso, alguno se anime a mandarnos su propia ficha del área que domine, para que sus conocimientos puedan servir a todos.

En este número se trata por primera vez el tema de las Redes Neuronales, un *ingenioso* modelo matemático cuya aplicación inmediata es el reconocimiento de caracteres y figuras. No deje de leerlo. *Cómo y dónde hacer FTPs* es una guía para los usarios de la comunidad Internet, algunos sitios interesantes para *bajarse* en el día las últimas fotos del *meteosat* o dejarle un e-mail al mismísimo Bill Clinton (esto, en próximos números...) Hablando de redes, *Lantastic 6.0* es una opción económica y verdaderamente eficaz cuando se trata de conectar entre sí varios ordenadores. Nuestro experto en redes, la analiza en su banco de pruebas particular en este número.

Para finalizar un toque de *hardware* con el artículo sobre Procesadores Risc, con todo lo que hay que saber sobre el tema. Gracias y hasta el mes que viene.

MARIO DE LUIS



ABRIL 1995. Número 8
Es una publicación de



COMMUNICATIONS, S.R.L.

Editor
Antonio M. Ferrer Abelló
Directora Comercial
Carmina Ferrer
Director de Producción
Carlos Peropadre
Publicidad
Magdalena Pedreño Llorente

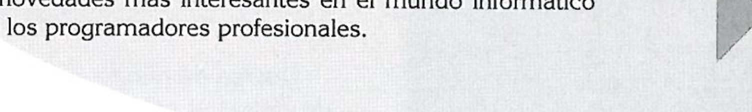
••••••

Director
Mario de Luis García
Redactor Jefe
Carlos Doral Pérez
Coordinador técnico
Miguel Angel Alcalde

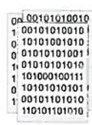
Colaboradores
María Gil, Juan Manuel Martín, Luis Martín, Agustín Guillén, Carlos Arias, Fernando J. Echevarrieta, David Brioso, Francisco G. Avilés, Francisco Monteagudo, Raúl Luna, David Aparicio, Emilio Postigo, Bernardo García, Ignacio Cea.

Edición técnica
Emilio Castellano
Maquetación
Fernando García Santamaría
Tratamiento de imagen
Josefa Fernández Martínez
Servicios Informáticos
Digital Dreams Multimedia, S.L.
Ilustraciones
Miguel Alcón
Secretaría de Redacción
Consuelo Jiménez
Suscripciones
Erika de la Riva
Redacción, Publicidad y Administración
C/ Marqués de Portugalete, 10
28027 MADRID
Telf.: 741 26 62 / Fax: 320 60 72
Filmación
Duvial
Impresión
G.D.B.
Distribución
MIDESA

Depósito legal: M-26827-1994
ISSN: 1134-4792



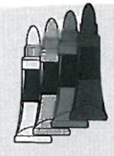




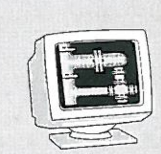








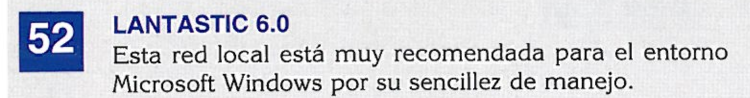




SUMARIO



Foto de las neuronas: Juan A. de Carlos, Miguel Morales, Juan Lerma. Instituto Cajal.

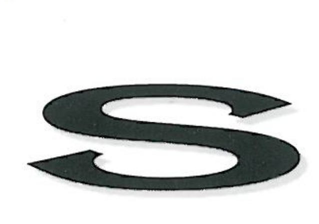


Sólo Porgramadores agradece al Instituto Cajal (C.S.I.C.) la cesión de las imágenes para la realización de la portada.





NOTICIAS



SISTEMAS AG DE SAP

La empresa alemana dedicada a la fabricación de aplicaciones de gestión SAP, ha anunciado la introducción en Iberoamérica de su sistema AG R/3. Antes que este, se había comercializado en España el R/2, destinado a la gestión empresarial en soportes mainframes.

El sistema R/2, en cambio, se dirige fundamentalmente a los sistemas cliente/servidor. Se ha enfocado a un nuevo mercado que comprende tanto a las compañías subsidiarias y afiliadas a las grandes corporaciones, como a las medianas empresas. Se ejecuta sobre sistemas UNIX de HP, DEC, IBM, SNI, BULL, SUN y, sobre todo, Windows NT de Microsoft. El R/3 soporta un concepto cliente/servidor de tres niveles, en los que los servidores de base de datos, de aplicaciones y de presentaciones, trabajan juntos vía LANs. Utiliza los interfaces gráficos más avanzados y las bases de datos relacionales más potentes.

EXPORACLE´95

El pasado mes de febrero tuvo lugar en el Palacio de Congresos y Exposiciones de Madrid la Segunda Edición de Exporacle. En ella se quiso ofrecer una visión completa de los productos y servicios Oracle. La exposición contó, además de con varios stands de Oracle, con los de aquellas empresas que colaboran con la anterior.

Entre los productos presentados por Oracle se encontraron algunos como el Sistema de Gestión de Bases de Datos R7.1, capaz de soportar aplicaciones críticas y escalables.En este sistema todo se "paraleliza" de forma dinámica: las consultas a la base de datos, la creación de índices, la carga masiva de datos y los backups. Esto permite el soporte de VLDB de centenares de Gygabytes.

Oracle mostró también el Workgroup Server, una potente base de datos relacional combinada con un acceso inmediato a herramientas Windows. Así, es posible contar con soluciones cliente/servidor para una amplia variedad de usuarios.

También hubo ocasión para el SQL TexRetrieval. Esta herramienta para el desarrollo de aplicaciones de Oracle permite integrar documentos



en aplicaciones comerciales multimedia basadas en el SGBD Oracle.

El resto de las empresas que se dieron cita en Exporacle también expusieron variados productos, siempre en relación con la tecnología Oracle.

Fujitsu introdujo su serie Nile, que ofrece altas prestaciones en arquitectura cliente/servidor. Ha sido diseñada para entornos de aplicaciones críticas en los que se requieren velocidad, disponibilidad de datos y escalabilidad del sistema.

Esta serie incorpora hasta 16 procesadores RISC basados en arquitectura de 64 bits. Soporta hasta 4 Gb. de memoria principal y hasta un 1 Tb. de almacenamiento en disco.

Otros expositores que participaron en esta feria fueron: Coopers & Lybrand; IBM; Sun Mycrosistems;

Informática El Corte Inglés, S.A.; Unisys; Hewlett-Packard Española, S.A.; Siemens Nixdorf; AT&T; y Compaq Computer; etc.

A las presentaciones de los expositores se sumó, como viene siendo habitual en este tipo de actos, un ciclo de ponencias que completaban teóricamente el evento. En este caso concreto, hubo multitud de ponencias, todas en torno al tema que vertebraba el evento.

BORLAND PRESENTA LA HERRAMIENTA DE CREACION DE BASES DE DATOS DELPHI

Borland, poco tiempo después de que Microsoft lanzase la última versión de FoxPro, ha puesto en la calle un serio competidor para esta herramienta. Se trata de Delphi, un potente "lenguaje" para la creación de bases de datos. Esta aplicación se ha rodeado de un entorno muy visual, compatible con Windows 95, que pretende facilitar al máximo la tarea de los programadores.

Una de las ventajas con las que cuenta este programa respecto de la versión de Microsoft es la de ser bidireccional. Es decir, todas aquellas acciones que se van realizando de forma visual, se traducen inmediatamente a código, y viceversa. De esta capacidad se deriva una mayor facilidad a la hora de crear aplicaciones. El código fuente que genera es PASCAL orientado a objetos, por lo que todos los usuarios de las versiones TURBO PASCAL de Borland se familiarizarán rápidamente con él.

BORLAND

Otra de las novedades de este producto es que no necesita intérprete, de modo que se compila de 10 a 20 veces más aprisa que con uno de ellos. No depende del módulo VBRUN.EXE, por lo que permite crear un solo ejecutable que contiene toda la aplicación.

Delphi permite trabajar con ficheros de base de datos comerciales de todo tipo, de modo que la reconversión de viejas bases de datos a esta nueva herramientas está asegurada al 100%. Es posible manejar cualquier tipo de datos, incluyendo ficheros de sonido y vídeos. También acepta periféricos variados.

Si se hiciese necesario añadir nuevas funciones a Delphi, pueden realizarse con otra herramienta, como Borland C++, para incorporarlas después en forma de librería DLL.

Para aquellos que quieran trasladarse de Visual BASIC a Delphi, existen traductores automáticos de un lenguaje a otro creados por otras empresas.

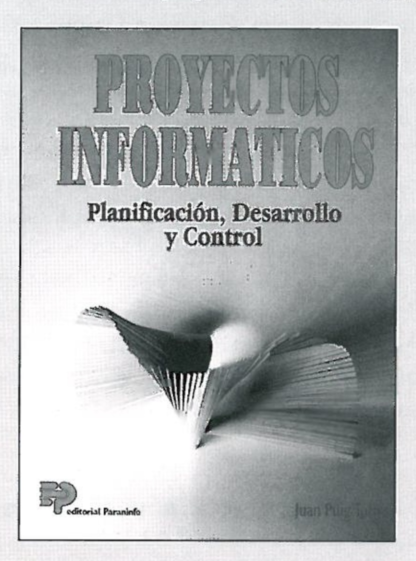
La versión cliente/servidor incorpora clases libres de derechos y listas para ser incorporadas en la aplicación de usuario.

Con Delphi se puede trabajar en red, y lleva incorporado un gestor de proyectos integrados. Estará disponible dentro de un mes, aproximadamente.

LIBROS

PROYECTOS INFORMATICOS. PLANIFICACION, DESARROLLO Y CONTROL

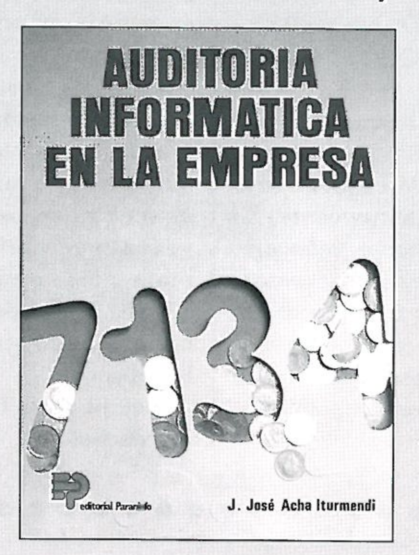
Predicando con el ejemplo, este libro aborda de forma sistemática y ordenada la planificación, puesta en marcha y conclusión de un proyecto informático. Dada la gran importancia que están alcanzando las actividades informáticas en los últimos tiempos, no se puede dejar nada al azar a la hora de realizar un proyecto de estas características. Por ello, para manejar con éxito aspectos de una compleja cadena tales como tareas, recursos, estrategia, planificación, modelo, técnicas de desarrollo, etc., nada mejor que una guía que vaya mostrando el camino.



Editorial: Paraninfo
Autor: Juan Puig Torné
122 páginas
Precio: 1.250 ptas.

AUDITORIA INFORMATICA EN LA EMPRESA

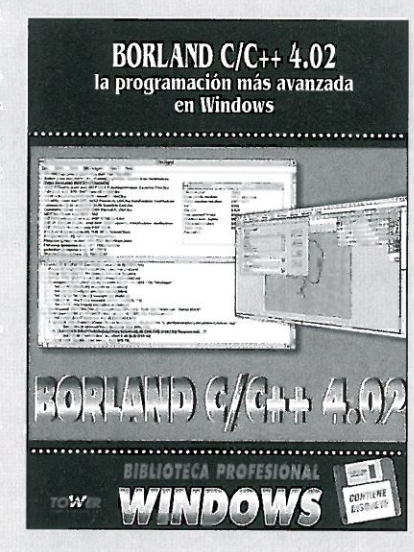
Tanto los profesionales de la informática como los directivos y responsables de las áreas de negocio de las empresas, podrán encontar en este breve volumen una información sencilla y completa. En el libro se han integrado tanto las aportaciones conceptuales como los casos prácticos necesarios para que se conozcan los conceptos básicos, los objetivos, las técnicas, las herramientas y los métodos relacionados con este aspecto. Se hace una exposición de los tipos de auditoría informática posibles, así como un repaso por las materias susceptibles de ser auditadas informáticamente.



Editorial: Paraninfo
Autor: J. José Acha Iturmendi
177 páginas
Precio: 1.560 ptas.

BORLAND C/C++ 4.02. LA PROGRAMACION MAS AVANZADA EN WINDOWS

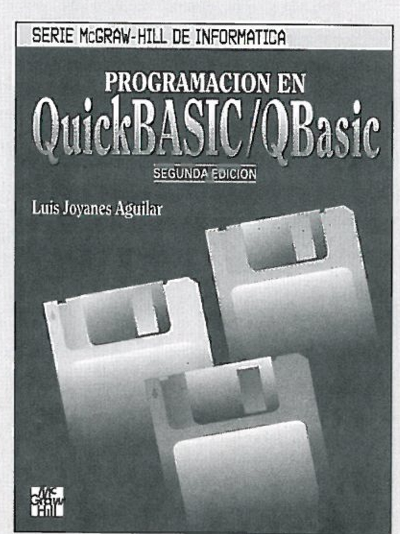
La alta programación en Windows tiene en Borland C/C++ uno de sus mejores aliados. Aunque este entorno cuenta entre sus ventajas con una gran facilidad de uso, nunca está de más un libro que guíe al usuario por el programa con paso firme. Algunos de los temas que se explican a fondo en este volumen son: las novedades más importantes que encierra Borland C/C++ 4.02, el entorno de desarrollo, el gestor de proyectos, el trabajo con AppExpert y ClassExpert, etc. Contiene disquete.



Editorial: Tower Communications, S.R.L.
Autor: Javier Guadalajara Loza
192 páginas
Precio: 1.695 ptas.

PRGRAMACION EN QUICKBASIC/QBASIC

Los principiantes y usuarios de QuickBASIC/QBasic podrán encontrar en esta completa guía una visión general de la programación estructurada con el compilador QuickBASIC y QBasic. La primera parte del libro se dedica a la exposición de los temas más importantes de la programación, y a cómo se trabaja con este lenguaje y sus elementos. El siguiente apartado se centra en la programación modular y los conceptos de funciones, subrutinas y procesos de archivos.



Editorial: McGraw-Hill
Autor: Luis Joyanes Aguilar
641 páginas
Precio: 4.915 ptas.

BESTIARIO

Hoy termina uno de esos días en que, citando a Mafalda, "lo peor de uno mismo son los demás". A las nueve veintisiete, mi jefe me ha declamado ese poema épico tan popular cuyo primer versículo empieza con "Que sea la última vez...", que se despliega posteriormente en cualquiera de las múltiples variaciones que ofrece la rica tradición oral española, y que invariablemente concluye con aquello de "...y si no te gusta, ya sabes dónde está la puerta". A las once treinta y cinco, mis compañeras me han sometido a público oprobio y baldón a cuenta del innovador diseño de mi corbata. A las dos veintiuno, he recibido nota de Administración en la que tienen el placer de comunicarme que me han subido el sueldo un cero coma siete por ciento, que es el guarismo de moda. Y a las cuatro cincuenta y ocho, mi impresora se ha transmutado en una trituradora de documentos y me ha manchado la camisa de tóner (y estoy harto de tanto frotar).

Después de un día así, yo exijo una satisfacción. Y me la voy a dar aquí y ahora. Hace tiempo, juré que la próxima vez que el mundo mostrara conmigo tan regocijada y gratuita crueldad, yo contraatacaría empezando a escribir mi bestiario particular. Una caja de Pandora donde encerrar a todas las personas, animales, y entes hardware/software que me ponen de los nervios. Es una cuestión de autodefensa, vaya. Y voy a empezar ahora mismo con las primeras fichas.

El brujo de la tribu

El brujo de la tribu es ese programador que sólo sabe hacer una única cosa, pero una cosa tan intrincadamente oculta en la telaraña del desestructuradísimo sistema de trabajo de la compañía que sólo él sabe cómo sacarla adelante. Además, construye un halo de misterio, humo y espejos, alrededor de aquello, de modo que nadie pueda nunca sospechar que en el fondo es más sencillo que el mecanismo de un chupete. Así, cuando uno tiene que vérselas con ese chisme, acude al brujo de la tribu, el cual, con gesto grave, ladea la cabeza, se acaricia la barbilla, dice "mmm, no sé, si se va a poder...", y acaba dando dos pases mágicos, pronuncia unas cuantas claves cabalísticas, te hace un parche y se va diciendo que le debes un favor, no lo olvides. El jefe le llama "el genio". Lo que sus compañeros le llaman es impublicable.

El solipsista de la cola de impresión

Los solipsistas practican la actitud filosófica de pensar que todo lo que se percibe es una ilusión, que no hay algo llamado realidad sensible, y que lo único que existe en el Universo son ellos mismos. Los demás somos alucinaciones. Eso deben de pensar los solipsistas de la cola de impresión, que se creen únicos usuarios de la impresora de la red, por lo que pueden mandar trabajos de gran extensión y luego olvidarlos. Un ejemplo. Mi jefe dice "¿Dónde &#*% está el informe que te pedí?". Yo contesto "Te lo di ayer". El replica "Pues dámelo otra vez". Yo gimoteo "Pero..." El se reafirma "Lo quiero antes de que este bolígrafo llegue al suelo". Y suelta el bolígrafo, yo alcanzo mi pecé en dos zancadas, abro el procesador de textos, selecciono "Imprimir", corro a la impresora de red, la impresora está detenida con un mensaje "Sustituir papel bandeja A4", yo abro el cajón del papel, el cajón del papel está vacío, corro al pasillo, vandalizo la fotocopiadora para conseguir un taco de folios, vuelvo en plena fibrilación ventricular, meto el taco de folios en la impresora, pulso "En línea", y en la bandeja de salida empiezan a aparecer las hojas del interrumpido y olvidado trabajo de ciento cincuenta folios del solipsista de la cola de impresión: veintiuno... veintidós... veintitrés... No, no está usted solo. Mi mujer asegura que a ella también le pasa.

El jeroglificocríptico

Este programador ha descubierto cómo asegurarse un trabajo estable por el resto de sus días. Muy sencillo: Haga programas incomprensibles, innecesariamente complejos, no ponga ni un comentario y jamás escriba ni una línea de documentación, aunque sea en una servilleta de papel. No escriba siquiera las entradas y salidas de su módulo. Esto último comuníquelo verbalmente, y enfádese si le piden que lo repita. Esta fórmula es mejor que un contrato blindado. Al cabo de unos meses, el jeroglíficocríptico es imprescindible en la compañía. La estructura de flujo de una rutina suya es lo más parecido a una piscina de espaguetis. Intentar leerla es como intentar separar la leche del Nesquik. Pretender mantenerla, como entrar en uno de esos templos plagados de trampas de las películas de Indiana Jones: si soplas, se te derrumba encima y te aplasta. Su pasatiempo: redactar programas de tres líneas del tipo "¿a que no sabes qué hace?". Su algoritmo favorito: cuatro funciones de quinientas líneas llamadas "Ordena", "Ordenar", "RealizaOrdenacion", y "Sort". Todas llaman a todas.

Bueno, por lo pronto allá van tres registros para mi base de datos. Ya me siento un poco mejor. La próxima vez que tenga un día plagado de estos virus humanos, los pincho con un alfiler y los añado a esta galería de personajes disecados.

ENTREVISTA

DIGITAL DREAMS MULTIMEDIA



Esta joven empresa es actualmente una de las más pujantes y creativas de España en el campo de la informática personal. Sus actividades abarcan desde la realización de software multimedia hasta la creación de videojuegos empleando las herramientas de desarrollo más potentes del mercado.

Durante la entrevista que Sólo Programadores mantuvo con DDM, destacó, sin duda, la vitalidad y juventud de sus componentes, así como su eficacia a la hora de afrontar nuevos retos. También hay que señalar el apoyo y promoción que DDM está dando a programadores independientes, ofreciéndoles tecnología y asesoramiento para sacar a la luz sus proyectos y aplicaciones, tanto en España como en el resto del mundo. Aunque no pudieron estar todos presentes, sí se contó con algunos de los miembros más característicos e interesantes de esta empresa: Daniel Navarro, Emilio Castellano, Luis Fernandez y Gabriel Hortas.

Toda aventura tiene un comienzo, ¿cual fue el origen de esta empresa?

El origen fue un grupo de amigos universitarios que en un momento dado deciden arriesgarse a vivir de la informática y crean una empresa propia para ser los propietarios de sus productos, con un espíritu de independencia. El año en que comenzamos fue el 84 y algunos juegos que hicimos en esta primera época fueron: "Nuclear Bowls", "Star Bowls", "Brick", "Oberón 69", "Jackson City", "Drakkar", "Fórmula 1", "Chesshousers" y "El Viaje de Colón", este último creado para la Asociación del V Centenario del descubrimiento de America.

Dentro del mundo informático, ¿en qué áreas trabajáis?

DDM es una empresa de servicios informáticos que se dedica a hacer productos multimedia, videojuegos, juegos interactivos para televisión y servicios informáticos a editoriales.

Hay varias empresas españolas dedicadas a desarrollar juegos y la competencia es muy seria, ¿cual es vuestra estrategia para obtener tan buenos resultados?

Creemos que por el momento DDM es el lider del mercado de los quioscos por la producción y por la calidad fundamentalmente. Hay que recordar que estamos produciendo un título al mes para su venta en quioscos de prensa,

grandes almacenes y tiendas de informática. Otro factor que influye es la variedad, tocamos todos los temas: aventuras, deporte, arcade, conversacionales, 3D... En el extranjero, *Epic Megagames* y *Apogee* distribuyen nuestro software por todo el mundo, pero siempre menteniendo nuestro sello.

Estamos intentando ser en cuestión de dos meses, la primera empresa española que consiga poner programas en los primeros puestos de las listas de venta del Reino Unido y de EE.UU. Este es nuestro objetivo y esperamos conseguirlo.

Parece que existe una tendencia a centrar los juegos en máquinas dedicadas como las que fabrican Sega, Nintendo o Sony. ¿En qué medida se pueden ver afectadas las empresas de vídeo-juegos para PC?

Es una realidad ineludible que las consolas se han hecho un hueco bastante cómodo en los hogares, porque la gente prefiere tener máquinas sencillas en las que, con sólo insertar un disco o cartucho, funcione el juego. Creemos que cualquier empresa que se precie tiene que abarcar el mundo del PC, de las consolas y también el de las máquinas recreativas.

Nosotros estamos estudiando las posibilidades de los últimos diseños de Sega, Nintendo y Sony (Saturn, Ultra 64 y PlayStation), así como la consola 3DO, para traducir nuestros juegos de PC a versiones arcade específicas para estas máquinas.

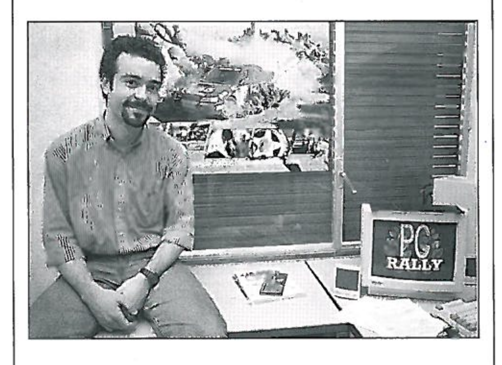
¿Que productos, programas, ha realizado DDM hasta la fecha?

En primer lugar hemos hecho programas interactivos para Tele 5: "Penalti" y "Taponazo". Ahora estamos desarrollando otro juego para las cadenas autonómicas "Balloon Baby" con un personaje propio de dibujos animados. Además tenemos una línea de software de entretenimiento como "USA Soccer", "PC Liga", "Super Quinielas", "Pack Opera 25", "Strip Poker", "Rol Crusaders" y "PC Rally" para venderlos en quioscos dentro de una línea de productos de consumo masivo. La otra rama, con un caracter más serio, es el servicio que damos a empresas y editoriales como por ejemplo, la realización de programas a medida para CD-ROM, aplicaciones multimedia, presentaciones o libros interactivos. Hasta la fecha, algunos de nuestros principales clientes han sido, la editorial Planeta, Ediciones Multimedia, Tower Communications o el Grupo Zeta.

Las aplicaciones multimedia también están diseñadas para venderlas al extranjero. De hecho se están traduciendo a 5 idiomas

Para llevar a cabo tantos proyectos es necesaria una organización muy sólida, ¿cómo está estructurado DDM?

En DDM cada persona es responsable de un área, y en todo momento existe una comunicación fluida entre todos los miembros para mejorar día a día las producciones. Tenemos una serie de coordinadores y responsables de áreas concretas. La unión y engranaje de todos los miembros de DDM la realizan Daniel, *director de proyectos;* Gonzalo Martín, *director técnico;* y Emilio Castellano, *coordinador de personal.*



Además contamos con Miguel Alcón, grafista y dibujante; Alberto, técnico en tratamiento de sonido e imagen; y Miguel Angel, encargado de documentar todos los proyectos;

Por último en la fase de comercialización, Carlos es el responsable de la producción y Mario, es el que lleva toda la gestión comercial.

Las aplicaciones multimedia requieren manejar una gran cantidad de recursos que complican el trabajo del programador, ¿en qué medida facilitan el desarrollo de aplicaciones los lenguajes de autor?

Para desarrollar determinadas aplicaciones, los lenguajes de autor son idóneos. Fundamentalmente han sido diseñados para crear libros interactivos, para fundir imágenes, animaciones, textos y sonido en un sólo paquete de software. Desde este punto de vista, lenguajes como Icon Author, Toolbook, Director, Ultimedia Builder, son muy adecuados para la realización rápida de software multimedia. Lo bueno que tienen es que la programación en estos entornos es muy sencilla, y en muy poco tiempo se consiguen resultados espectaculares.

Sin embargo, cuando se requieren funciones más complejas, como manejar un volumen excesivo de datos o acceso a funciones muy específicas, este tipo de herramientas se quedan cortas, por lo que se tiene que recurrir a la programación tradicional.

Los nuevos sistemas operativos como Windows 95, ¿limitarán la creatividad de los programadores?

Windows 95, si no tuviera la opción de ejecutar el DOS en modo real y de desinstalarse de la memoria, desde luego que limitaría bastante la libertad y creatividad, porque a pesar de que existen ya librerías para desarrollar juegos bajo Windows, el sistema operativo impide acceder al hardware directamente, por lo que recursos como por ejemplo, modificar los registros de la VGA, en Windows 95 no serían posibles. La programación de juegos en entorno Windows queda relegada a aquellos que no necesiten aprovechar al máximo los recursos de la máquina.

Los continuos cambios en los ordenadores personales, como Plug and Play, las nuevas arquitecturas internas, el aumento de la velocidad de proceso, ¿cómo afectan a vuestra labor de desarrollo?

En principio no tienen por qué perjudicar los avances, aunque sí se requiere poseer un información actualizada de los equipos existentes en el mercado. De hecho, se cumple el siguiente dicho: "No es mejor programador el que mejor lo hace, sino el que antes se entera de dónde encontrar la rutina o documentación que necesita". De esta forma se ahorran muchas horas de trabajo.

¿Cómo surge una idea para un juego?

De muchas maneras, la mayoría son fruto de la casualidad, de ir andando

por la calle y fijarte en un detalle tonto, también de la inspiración en trabajos anteriores. Y sobre todo, de ver y usar mucho otros prtogramas. Un programador de juegos tiene que ser un *jugón* profesional para obtener nuevas ideas y métodos.

¿Cómo se crea crear un juego y cual es la fase más importante?
Nace a partir de una idea que proponemos o que nos proponen. Al principio el dibujante crea el diseño de los personajes y otra persona realiza el guión y el *storyboard.* El director de proyectos distribuye este material entre los programadores. Poco a poco se va refinando el proyecto añadiendo nuevos medios y mejorando la definición de los dibujos. Durante todo el proceso, una serie de supervisores van comprobando la jugabilidad, dan nuevas ideas para mejorarlo y antes de acabarlo lo adaptan a los distintos entornos donde vaya a funcionar. Una vez acabado se modifica para enviarlo a otros países, traduciéndolo y transformándolo en un arcade convencional.

Lo más importante de todo es coordinar los esfuerzos de los programadores y las exigencias comerciales. Es muy importante determinar los objetivos de cada persona. También hay que lograr que la jugabilidad y diversión del juego satisfagan los requerimientos del usuario final. También son muy importantes las opiniones de los probadores o *testers* que aconsejan a los programadores.

Los primeros juegos es seguro que los programasteis en ensamblador debido a las limitaciones de las máquinas. Pero ahora esta dificultad debe estar ya superada. ¿Cómo habéis vivido la evolución de la informática personal?
Esta es una pregunta que cualquier programador que lleve en la profesión más de 12 años estará encantado en contestar. Y nosotros no vamos a ser menos.

En los tiempos del Spectrum, en 1982, un programador solo, en su casa, con un ordenador de 8 *bits* y con ciertas dotes artísticas, podía crear el código, gráficos, argumento e incluso hasta la música de un juego. Ahora en nuestros días esto es impensable.

Mirándolo objetivamente los medios no han cambiado tanto, los que sí lo han hecho son los métodos. Los programadores siguen siendo los mismos aunque con más experiencia, pero ahora el trabajo en equipo ha cobrado una importancia a la que nadie puede dar la espalda. No es extraño ver en los créditos de cualquier aplicación multimedia o juego, a 10 ó 15 personas. En el 82 bastaban una o dos personas y tres ya eran multitud.

¿Cuál es el lenguaje de programación que empleais para resolver puntos críticos, como por ejemplo la velocidad?
Los algoritmos se suelen hacer en C por comodidad, por la teoría que ya hay acumulada y como forma de intercambio de los métodos de programación. Sin embargo, cuando se trata de rutinas críticas como poner un sprite, realizar scrolles, rutinas de teclado, ratón, etc, lo ideal es codificarlas en ensamblador e incluirlas en una librería para así poderlas usar más adelante.

Para enlazar todas las partes de un programa, ¿utilizais algún compilador en concreto?
Hasta hace poco hemos estado empleando Borland C++ porque es el que tradicionalmente hemos preferido. Ahora hemos empezado a trabajar con Wattcom C++, debido principalmente a que permite programar en 32 *bits* bajo DOS. Esta decisión se ha tomado porque pensamos que el ordenador que domina el mercado de PC instalados es como mínimo un 386, y al programar especificamente para este procesador se aprovechan mejor los recursos disponibles, mejorando el rendimiento de las aplicaciones.

¿Qué librerías, herramientas y métodos empleáis para producir los proyectos?
Para rentablizar el trabajo de las personas contamos con librerías propias de gráficos en 2 y 3 dimensiones, sonido, etc., que se actualizan constantemente y que están diseñadas para su empleo en una plataforma C-PC. A parte de desarrollar juegos y aplicaciones hay gente creando nuevas librerías. Estos elementos nos permiten sacar en tres meses proyectos que normalmente llevarían más del doble.

Esto repercute en que si lo hemos hecho nosotros nos sale más barato, hemos optimizado el desarrollo, y si es un *freelance* el que lo hace, cobra más por cada mes trabajado.

En este aspecto quizás seamos la empresa que más cuida el método de trabajo de sus empleados. A efecto de plantilla esto se plasma en un coordinador por cada área y siempre unos programadores y grafistas estrella para resolver cualquier problema. Los programadores que tenemos son posiblemente de los mejores grupos que haya en nuestro país actualmente, porque llevamos 11 años realizando proyectos.

Con toda esta experiencia nos es muy fácil poner en marcha cualquier cosa, desde un juego de golf hasta un programa multimedia donde un personaje hable y se mueva.

¿Cual sería la característica más importante que diferencia la programación de un juego respecto a otro tipo de desarrollo?
Lo mejor de los vídeo-juegos es que el límite para hacerlos son la imaginación y el tiempo que se disponga para acabarlo.

Los efectos más descabellados y las transformaciones más alucinantes pue-

Digital Dreams Multimedia produce una línea de software de consumo masivo para su distribución en quioscos, tiendas de informática y centros comerciales. Durante el mes de Marzo saldrá al mercado PC Rally.











den hacerse realidad en un juego. Lógicamente, nadie espera que en una base de datos existan sprites que se paseen por la pantalla o tenga efectos sonoros de disparos o explosiones al borrar un fichero.

Lo esencial de programar un juego es la creatividad, y quizás podrían considerarse como una obra artística los juegos que consiguen divertir y enganchar a los usuarios envolviéndolos en una especie de halo mágico.

Poca gente que ha tenido la experiencia de hacer un vídeo-juego luego ha sido capaz de resistir la tentación de seguir haciéndolo durante un buen número de años. Nosotros somos la mejor prueba de ello.

¿Cual es el ritmo de trabajo mientras se programa un juego?
No hay un horario fijo, de hecho puedes despertarte de madrugada para escribir una idea genial, o bien puedes pasarte dos días sin programar para pasar a estar una semana durmiendo 4 escasas horas diarias. Depende mucho de la inspiración. No es como programar bases de datos que es bastante monótono al depender más del método que de la creatividad, bueno, es sencillamente diferente.

Lo extranjero siempre se ha considerado mejor que lo nacional. ¿A qué se debe esto? ¿Están los programadores españoles preparados para competir en el mercado americano?
En España hay programadores muy buenos y no es de extrañar. Existe una frontera impuesta por el tipo de tecnología empleado, que, una vez alcanzada, no se puede superar.
Mientras no cambie la arquitectura física de los ordenadores (empleo de varios procesadores, comunicaciones, nuevos periféricos específicos para realidad virtual), no se puede seguir avanzando.

La diferencia estriba en que las empresas americanas cuentan con más medios. Hace poco vimos un juego en el que aparecía Denis Hooper, un conocido actor de películas de terror. Las empresas europeas aún no se pueden permitir esto. En una producción es imposible contratar a actores de categoría internacional.

Hace poco recibimos la noticia de que Steven Spielberg había creado una división para crear software recreativo y la había dotado con un presupuesto de 75 millones de dólares. Lógicamente, con un presupuesto así ya no es cuestión de creatividad o de conocimientos, depende únicamente de los medios.

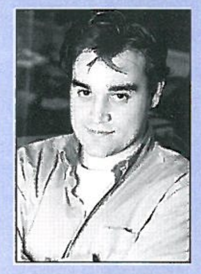
¿Qué formación debe tener una persona para dedicarse a programar juegos?
Lo primero que se necesita es cierta afición, aunque es muy importante la formación. Nuestro prototipo de programador sería el siguiente:

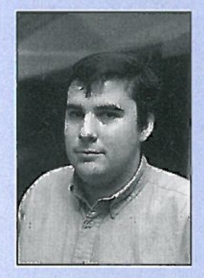
- Estudios universitarios (carrera técnica)
- Buenos conocimientos de C, C++ y por supuesto, ensamblador.
- Dominio de alguna característica de los ordenadores: gráficos, sonido, programación hardware de la VGA o tarjetas de sonido, etc.
- Experiencia en el desarrollo de algún juego.

Para nosotros es muy importante la formación universitaria, ya que permite abarcar temas complejos como la programación en 3D, en donde la física y las matemáticas resultan imprescindibles. Por ejemplo, el algoritmo del *Ray Casting* empleado para generar los paisajes en 3D de juegos como *DOOM*, emplea árboles de búsqueda binarios (B.S.T.) y que se estudian en el tercer o cuarto curso de una carrera universitaria de Informática.

¿Que les recomendarías a los programadores que quieren realizar juegos o programas multimedia?
Que se pongan en contacto con nosotros, porque les vamos a ofrecer una infraestructura que no puede tener una persona que trabaja por su cuenta. En nuestra empresa contamos con una enorme librería de rutinas y de medios para sacar adelante cualquier proyecto. Tenemos varios PC 486 y Pentium, 6 Macintosh Quadra, escáners de mesa, digitalizadoras de vídeo, sonido, grabadora de CD-ROM, acceso a Internet, en resumen, todos los medios para hacer el producto completo. Por supuesto damos las mayores garantías en comercialización y distribución. No nos limitamos a hacer el software, todo lo que le acompaña, manuales, tableros, es realizado por un equipo de maquetación, de forma que se consigue realizar todo en DDM.

Algunos componentes de DDM.







Por último, ¿que tenéis preparado para dentro de unos meses?
Lo fundamental es vender en el extranjero, lograr situarnos en los primeros puestos de las listas de venta y conseguir prestigio a nivel mundial. Esto lo haremos directamente nosotros, mediante cauces de distribución normales: shareware o distribuidoras, pero manteniendo nuestro sello, ya que lo hemos hecho así durante toda la vida.

También vamos a crear versiones arcade de nuestros juegos para máquinas recreativas y consolas. Estamos obteniendo licencias para desarrollar sobre *Sony PlayStation*, *Nintendo Ultra64* y *Sega Saturn*. De esta forma estaremos también presentes en el desarrollo de aplicaciones multimedia para este tipo de plataformas.

Lo último que estamos haciendo es un programa de Tenis para el cual estamos buscando una licencia importante

Miguel Angel Alcalde.

INTERRUPCIONES EN MS-DOS

Emilio Postigo

001010100010
00101010010
10101001010
01010101001
01010101010
101000100111
10101010101
00101101010
110101101010

En este capítulo vamos a hablar de las interrupciones. Éstas son subrutinas suministradas por el sistema operativo o por la ROM del procesador, que permiten efectuar con comodidad tareas comunes (aceptar una tecla, imprimir un carácter, leer o escribir un sector de un disco, etc), sin necesidad de que el programador deba acceder por su cuenta a los puertos necesarios para efectuar esas labores. Sin las interrupciones la programación en ensamblador sería más dificultosa de lo que ya es.

En los primeros capítulos del curso de ensamblador se habló de los recursos del sistema. Asimismo han ido apareciendo referencias continuas a los mismos, como prueba de que no es posible pensar en la programación de dispositivos o a bajo nivel sin emplearlos continuamente. En este artículo se condensará lo visto hasta el momento, y se introducirán nuevos conceptos. Para ello, se comenzará estudiando las interrupciones desde el punto de vista de su utilidad, es decir, como procedimientos a disposición del programador. Una vez comprendidas ciertas ideas fundamentales, se pasará al estudio de las interrupciones desde el punto de vista de su activación.

La existencia de las interrupciones hace viable la programación en ensamblador. Son las herramientas de más alto nivel con que puede contar un programador, antes de construir las suyas por medio de procedimientos, y constituyen la base de cualquier interacción entre la CPU y el exterior.

LA INTERRUPCIÓN COMO PROCEDIMIENTO

En los dos capítulos anteriores vimos cómo se invocaba una subrutina mediante la instrucción CALL. En una primera aproximación, que es la que interesa en la mayoría de las ocasiones, una interrupción es un procedimiento más a disposición del programador que debe ser invocado mediante la instrucción INT nº-, donde nº indica la interrupción. Los procedimientos habituales eran llamados con la instrucción CALL, seguida de una etiqueta (llamada directa), o una serie de posiciones de memoria (llamada indirecta). Que la interrupción sea invocada mediante un número significa que, de alguna forma, el procesador es capaz de encontrar el código inicial de ésta a partir de ese número. Esto es posible debido a la existencia de la llamada tabla de vectores de interrupción (TVI). Esta tabla se inicializa cada vez que se enciende el equipo, y en ella se almacenan las direcciones físicas de todos los tipos de interrupciones que puede manejar el procesador. Su número asciende a 256, de la 00h a la FFh, siendo habitual escribir el número de interrupción en hexadecimal. Dado que cada dirección física ocupa 4 bytes, la TVI ocupa en total 4 * 256 = 1024 bytes. Cuando el procesador funciona en modo real o virtual 8086, esta tabla se localiza a partir de la dirección física 0000:0000, y finaliza en la 0000:03FF. Como es de esperar, los vectores de interrupción se almacenan consecutivamente, ocupando el de la interrupción 0 los 4 primeros bytes de la tabla. Los cuatro siguientes corresponden al de la 1, etc, pudiéndose localizar cualquiera de ellos mediante el cálculo 4 * n. La figura 1 ilustra el contenido de esta tabla.

La TVI comienza en la dirección física 0000:0000

Cuando una instrucción INT nº es ejecutada, a nivel de procesador ocurre lo mostrado en la figura 2. Se decrementa SP para hacer espacio al valor

actual del registro de flags. A continuación TF e IF son puestos a 0. Después se pone en la pila CS y se inicializa con el segmento de comienzo de la interrupción, extraído de la tabla de vectores. Se guarda también el desplazamiento de la instrucción que se ha de ejecutar tras la interrupción, y se inicializa IP con el desplazamiento de comienzo del código de la interrupción. Este código debería ser llamado gestor o manipulador de interrupción, pero es habitual emplear la palabra interrupción a secas. El código gestor finaliza con la instrucción IRET, que saca de la pila, como se ve en la figura 3, todo lo introducido por INT. Obsérvese que, salvo la introducción de los flags, el orden del almacenamiento en la pila producido por INT, coincide con el de la instrucción CALL FAR. Esta característica será utilizada al llegar al tema del cambio del vector de una interrupción.

La instrucción INT pone a 0 los flags IF y TF

FUNCIONES Y PARÁMETROS

Existen interrupciones que efectúan una única tarea. Por ejemplo, la interrupción 05h sólo vuelca la pantalla en la impresora, la 20h se limita a finalizar un programa y devolver el control al sistema operativo, etc. Sin embargo, es habitual que una interrupción efectúe más de una labor, hablándose por lo tanto de las distintas funciones de ésta. Cuando el programador requiere los servicios de una función concreta de una interrupción, debe depositar en el registro AH el número de esa función. Así, por ejemplo, si se va a emplear la interrupción 21h para visualizar una cadena de caracteres, previamente se debe haber movido a AH el valor 09h, para seleccionar la función que se encarga de esta tarea. Si hubiéramos necesitado que el usuario introdujese una cadena de caracteres por teclado, habría sido necesario inicializar AH con el valor 0ah.

Este sistema permite suministrar distintos servicios sin consumir más de un vector de interrupción. Por ejemplo, la interrupción 21h efectúa alrededor de 130 tareas diferentes, que se presentan en forma de funciones habitualmente llamadas funciones del DOS.

Ahora bien, en la mayoría de las ocasiones es necesario no sólo indicar qué servicio es requerido, sino también a partir de qué datos es necesario trabajar: es decir, es necesario pasar parámetros a una interrupción. Por ejemplo, si se va a visualizar una cadena, es preciso indicar dónde comienza ésta. Si se desea abrir un fichero, habrá que especificar su nombre, la unidad en la que está, el modo de apertura, etc.

Al estudiar los procedimientos se vio que los parámetros de una subrutina eran pasados a la pila en un orden fijo "convenido" con la misma. El código de la subrutina se encargaba de recuperarlos, depositándolos en registros. Esta forma suele ser la más práctica a la hora de diseñar subrutinas estándar, que van a estar en librerías y van a poder ser utilizadas por diferentes programas escritos en distintos lenguajes. Sin embargo existen métodos más rápidos (pero que plantean otros problemas) para este fin, como por ejemplo el empleo de áreas comunes de memoria (variables globales), y el uso de registros. Como ya se ha podido ver en diferentes programas de ejemplo, los parámetros se pasan a una interrupción mediante este último sistema. Así, en el caso de la función 09h, el desplazamiento de la cadena que se deseaba visualizar era movido a DX. Sólo se emplea la pila para almacenar los flags y la dirección de retorno. Eso sí, no se olvide que el número de registros es limitado. Si es necesario pasar muchos parámetros, lo usual es emplear para ello la dirección de una estructura, de formato predefinido.

Una interrupción puede devolver valores al programa principal, bien como datos que pueden ser utilizados, bien como señal de que el proceso ha concluido con éxito, o no. Para lo primero se emplean registros (AX, AL, AH, DL), y para lo segundo el flag CF. Éste vale 0 cuando la operación finaliza satisfactoriamente, y 1 en caso contrario. Si es así lo usual es que se devuelva también en AH el código del error que se ha producido. Como rasgo a tener en cuenta cuando se codifican interrupciones propias, si se desea devolver información a través de los flags, el cambio ha de efectuarse sobre la zona de la pila que las guarda, dado que éstas se almacenan en la llamada y se recuperan mediante IRET.

Hasta el momento, hemos visto cómo funciona el mecanismo de llamada de la CPU al código gestor de una interrupción. Ahora bien, el modo en que aquélla determina cuándo debe ser invocado un proceso u otro permite distinguir tres tipos de interrupciones: hardware o externas, excepciones, e interrupciones internas o software.

INTERRUPCIONES HARDWARE

Son aquellas que tienen lugar como respuesta a la solicitud de un elemento externo al procesador. Estas interrupciones permiten la comunicación entre los periféricos y el procesador, y se subdividen a su vez en dos tipos: enmascarables y no enmascarables. Las primeras son aquellas cuya solicitud llega al procesador a través de la patilla INTR. Son peticiones "normales" de los distintos periféricos, solicitando un servicio a la CPU, que ésta suministra mediante el gestor apropiado. De esta forma se puede introducir una tecla en el buffer de teclado, actualizar el contador de la hora, etc, disponiendo de un interfaz adecuado entre el procesador y el "exterior".

Cuando la CPU recibe una petición por la línea INTR, comprueba el valor del flag IF. Si ésta vale 1, se finaliza con la instrucción que se esté ejecutando en ese momento, se identifica el periférico que efectúa la solicitud y se lanza el gestor apropiado. Si el flag IF vale 0, el procesador ignora la solicitud de interrupción, y se dice que las interrupciones están enmascaradas. El programador puede poner a 0 el flag IF empleando la instrucción CLI, y a 1 mediante STI.

Estas interrupciones pueden además ser inhabilitadas de forma parcial o individual, usando las posibilidades del controlador de interrupciones 8259. Este chip se encarga de controlar 8 interrupciones, enmascarándolas, asignándoles prioridades, y avisando al

procesador de la existencia de las oportunas solicitudes de interrupción, pudiendo haber más de uno en una placa base. Para llevar a cabo esta labor, dispone de tres registros internos de un byte: IRR o registro de solicitud de interrupción, ISR o registro de interrupciones en activo, e IMR o registro de máscaras de interrupción. El primero de ellos tiene asignada la dirección de puerto 20h, y se encarga de registrar, poniendo el bit correspondiente a 1, las peticiones de las interrupciones hardware con IRQ comprendidas entre 0 y 7. De esta manera, un 1 en la posición 7 significa que hay una solicitud por parte de la impresora. El segundo registro almacena el estado de ejecución de los distintos gestores, de forma que antes de lanzar a la CPU la correspondiente petición de un periférico, se comprueba el estado y la prioridad de las restantes. Estas prioridades disminuyen con el número de IRQ, de forma que la 0 tiene mayor peso que la 1, y así sucesivamente. Por último, el tercer registro informa al chip de las interrupciones que, con un 1 en la posición adecuada, están desactivadas. Evidentemente, antes de pedir a la CPU la ejecución de una rutina, el 8259 comprueba el estado de la máscara correspondiente. Un 1 en la posición 1 de este registro significa que no se atienden peticiones del teclado por ejemplo.

En caso de tenerse otro chip controlador de interrupciones, las direcciones de puertos del IRR y el IMR son, respectivamente, A0h y A1h. Mediante éstos se gestionan las interrupciones de IRQ comprendidas entre 8 y 15. Las prioridades de estas interrupciones están intercaladas con las del chip anterior de forma que, la IRQ9 tiene menos prioridad que la IRQ2 y más que la IRQ3.

Las interrupciones no enmascarables son aquellas cuya petición llega a través de la patilla NMI (Non Maskable Interrupt). Este tipo de solicitud no puede ser ignorado, y está relacionado con situaciones graves dentro del sistema: caídas de tensión, errores de paridad en la memoria, etc. En la figura 4 se muestran las interrupciones que componen el grupo de las enmascarables. La NMI es una única instrucción y su vector es el 02h.

Las interrupciones hardware se pueden enmascarar individualmente

EXCEPCIONES

Las excepciones son procesos lanzados por la CPU a raíz de acontecimientos, detectados por ella, que impiden la ejecución de la instrucción siguiente. Situaciones de excepción son por ejemplo la división entre 0, la interrupción paso-a-paso, errores de cálculo del coprocesador, códigos de operación no válidos, etc. Por ejemplo, si se quiere comprobar el funcionamiento de una excepción, bastará con ejecutar las instrucciones MOV BX, 0 Y DIV BX. Al tener lugar una división en la que el cociente no cabe en el campo asignado a este fin, se activa el gestor cuyo vector de interrupción es el 0.

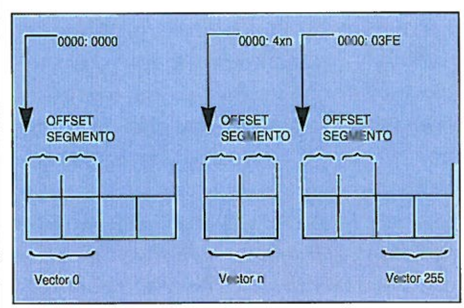


Figura 1.

Una excepción muy interesante, ya que permite la existencia de programas depuradores, es la llamada habitualmente interrupción paso-a-paso. Sabemos que la CPU trabaja en modo paso-a-paso cuando el flag TF toma el valor 1. Después de la ejecución de una instrucción, el procesador comprueba el valor de este flag. Si es 1, transfiere el control a una rutina cuya dirección de comienzo se encuentra almacenada en ese vector. Esta interrupción se trata como otra cualquiera, de manera que el flag TF vale 0 mientras se ejecuta.

Las excepciones suelen ser clasificadas por algunos autores, como casos particulares del siguiente grupo. En la figura 5 se muestran los vectores correspondientes a algunas excepciones.

INTERNAS O SOFTWARE

Las interrupciones internas son aquellas que se invocan desde el programa mediante la instrucción INT nº o INTO. El funcionamiento de la primera es ya conocido y sirve para ilustrar el de INTO. Esta instrucción, interrupción por overflow, produce una interrupción de tipo 4 si, en el momento de ser ejecutada, el flag OF vale 1. En caso contrario la interrupción no se ejecuta. ¿Para qué sirve esto?. Veámoslo: este flag se activa, cuando el resultado de una operación aritmética sobrepasa la capacidad de almacenamiento de un operando, si se trabaja con lógica de signo. Por ejemplo, supongamos que se efectúa la suma 7Fh + 01h. El resultado de esta operación (127+1) es 80h (128), que se debe interpretar de forma distinta según el programador haya decidido o no trabajar con lógica de signo. Si ha optado por lo último, a nivel de operandos máquina, se da la curiosa situación de que al sumar dos números positivos se obtiene uno negativo (80h = -128 bajo lógica de signo). Por lo tanto se dice, que se ha sobrepasado la capacidad de representación numérica de los operandos.

Para no cometer un error al propagar esta situación no deseada, el programador puede diseñar su propio gestor de interrupción por overflow, y asignar al vector 4 la dirección física del código que vaya a utilizar.

Las excepciones son procesos lanzados por el procesador

Para finalizar, decir que este tipo de interrupciones lo son sólo de nombre, ya que no interrumpen el funcionamiento de la CPU. Pueden ser consideradas como una llamada más a un procedimiento dentro de un programa, que permiten acceder a los recursos del sistema, mediante el mismo mecanismo



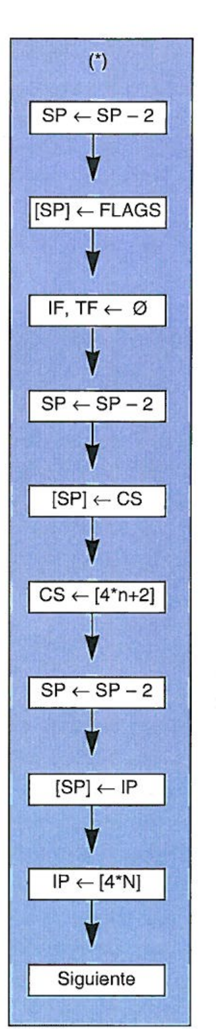


Figura 2. 'Fases en la ejecución de INT n'.

empleado por el procesador para invocar a los gestores de interrupción. Dentro de estas interrupciones, podemos incluir gran parte de las suministradas por la BIOS y el sistema operativo. En la figura 6 se muestran varias de las más usadas. Advirtamos que algunas de ellas, como la 23h y la 24h, no deben ser invocadas directamente, ya que en principio son usadas por otras tras cierta preparación. Asimismo, algunos vectores de interrupción almacenan la dirección física de una tabla de datos, como por ejemplo los vectores 1Eh y 1Fh.

La información mostrada para las interrupciones y sus vectores correspondientes puede variar de un sistema a otro, dependiendo del modelo de procesador y su modo de funcionamiento, real o protegido. En los textos mencionados en la bibliografía, se puede hallar información complementaria a este resumen.

CHOQUE DE INTERESES

Ahora bien, se ha visto que las interrupciones hardware y las excepciones se originan de forma impredecible para la CPU. ¿Qué ocurre si dos de ellas se producen a la vez?. En el caso de presentarse varias peticiones de interrupción o excepción simultáneamente, el procesador atiende a la de mayor prioridad. El resto queda "aparcado", hasta que finaliza la que se ejecuta en primer lugar, o hasta que ésta permite la activación de las pendientes o parte de ellas.

La mayor prioridad corresponde a la excepción de división entre 0, las interrupciones software activadas con INT y a la activada por INTO, por este orden. A continuación se atienden las interrupciones no enmascarables, las enmascarables si IF = 1 y, por último, la excepción paso a paso.

Así, por ejemplo, si en un mismo momento confluyen una interrupción software, una NMI y una hardware, primero se ejecutará la instrucción INT correspondiente a la primera. Antes de ejecutarse la primera instrucción del código de su gestor tendrá lugar la NMI y, a no ser que se active en algún momento IF, la interrupción externa deberá esperar a la finalización de éstas, antes de tener lugar.

MODIFICACIÓN DE INTERRUPCIONES

Una vez conocido el modo de activación y el funcionamiento de la llamada a un gestor de interrupción, lo que interesa al programador es poder manipular a su antojo las interrupciones existentes y crear otras nuevas.

En el primer caso trataremos la supresión de una interrupción o una función, así como de añadir una de estas últimas, de forma temporal o permanente. Veamos en primer lugar cómo se puede anular durante la ejecución de un programa la interrupción 05h. Esta es la encargada, como ya se indicó más arriba, de volcar la pantalla en la impresora. En los programas EJEMPLO1.ASM y EJEMPLO2.ASM se muestra una sencilla manera de desactivar esta interrupción durante la ejecución de un programa.

IP ← [SP]
SP ← SP + 2
CS ← [SP]
SP ← SP + 2
FLAGS ← [SP]
SP ← SP + 2
Siguiente

Figura 3. 'Fases en la ejecución de IRET'.

Supongamos por ejemplo que, por no tener conectada una impresora a nuestro equipo o por otros motivos, deseamos evitar cualquier problema en caso de pulsar la tecla <Impr Pant>. La solución es asignar al vector inicial de esta interrupción la dirección física de un fragmento de código que sólo se compone de la instrucción IRET. Una vez realizado el cambio del vector, cuando se pulse <Impr Pant>, el procesador sólo podrá efectuar la llamada a la interrupción y retornar de ella inmediatamente.

MANIPULAR LA TVI

En el programa EJEMPLO1 se sigue un proceso directo para efectuar el cambio del vector; como se puede observar, éste consiste en direccionar la zona de memoria ocupada por la TVI. Para ello se mueve el valor 0000h, la base del segmento en el que se encuentra esta tabla, al registro ES. Dado que no es posible mover directamente

La interrupción por overflow se asigna al vector 04h

una constante a un registro de segmento, se efectúa un paso intermedio colocando antes el valor 0000h en AX mediante la instrucción XOR. En SI se almacena el desplazamiento del vector con respecto a la base del segmento. En nuestro caso se tiene 5 * 4 = 20 = 0014h. A continuación se mueven a las variables Offset05h y Segmento05h los valores correspondientes del vector, para poder recuperarlos al finalizar el programa. Cuando estos valores están "a buen recaudo", se mueve a la parte baja del vector el desplazamiento que ocupa en el programa la instrucción IRET que va a sustituir al verdadero gestor. En la parte alta del vector se deposita el contenido del registro CS, ya que en este instante almacena el segmento en el que se encuentra IRET. Para este fin también hubieran podido emplearse los registros DS y SS, ya que se está tratando con un archivo COM, pero no ES: no se olvide que acabamos de modificar su valor para direccionar la TVI y no lo hemos restaurado.

Un detalle que conviene destacar es la forma en que se ha direccionado la zona de memoria correspondiente al

vector 05h mediante el operando ES:[SI]. Cuando se estudiaron los modos de direccionamiento se vio que, en función del tipo y los operandos, la CPU calculaba la dirección física de la posición de memoria correspondiente, tomando como base los registros de segmento DS o SS. En concreto, cuando se emplea el operando de memoria [SI], la CPU toma el valor de este registro, y lo suma con DS multiplicado por 16 (10h), obteniendo la situación exacta del byte con el que se desea trabajar. Podemos emplear el denominado prefijo de segmento ES, para direccionar con respecto al valor que almacena. El procesador dispone de cuatro en total, CS:, DS:, ES: y SS:, y tienen la capacidad de alterar la base de direccionamiento de cualquier operando de memoria. Su efecto no es permanente en modo alguno y se restringe a la instrucción en curso. Téngase presente que, aunque en un archivo fuente los prefijos acompañan al operando de memoria, en código máquina son situados antes de la instrucción.

INTERRUPCIONES EXTERNAS

Vector	IRØ	Empleo
Ø8 h	Ø	Temporizador
Ø9 h	1	Teclado
ØA h	2	Canal 1/0
ØB h	3	COM 2
ØC h	4	COM 1
ØD h	5	LPT 2
ØE h	6	Controladora disco flexible
ØF h	7	LPT 1
7Ø h	8	Reloj de tiempo real
71 h	9	Desvío a IRØ2
72 h	1Ø	Reservada
73 h	11	Reservada
74 h	12	Reservada
75 h	13	Coprocesador aritmético
76 h	14	Controladora HD
77 h	15	Reservada

Figura 4.

Tras este inciso continuemos con el programa ejemplo. Una vez efectuado el cambio del vector 05h, se ejecutaría el programa en sí. En este caso consta únicamente de la visualización de un mensaje por pantalla. Finalizado el programa, se accede a las variables que guardan la dirección del verdadero gestor, se restaura éste y se vuelve al DOS. Obsérvese que siempre que se ha escrito sobre la TVI, las interrupciones hardware se han desactivado con la instrucción CLI. Una vez finalizado el proceso de cambio, se han vuelto a habilitar con STI. Esto se hace para prevenir el caso de que la interrupción 05h se active con el vector a medio cambiar. Dado que ésta se invoca mediante la pulsación de una tecla, otra posibilidad habría sido enmascarar sólo la interrupción 09h, que es la interrupción hardware de teclado, utilizando las posibilidades del chip 8259. Este método tiene una ventaja: cuando se desactivan las interrupciones externas la CPU puede "perderse" llamadas importantes de los periféricos; por este motivo es aconsejable, si ello es posible, inhabilitar sólo aquellas interrupciones que puedan poner en peligro la ejecución de un fragmento delicado del programa.

Las interrupciones software poseen la máxima prioridad

En fragmentos de código como los que hemos empleado, la cosa no tiene mayor importancia. Sin embargo, cuando se trata de la ejecución de un gran conjunto de instrucciones, el método de enmascaramiento ha de ser un motivo de reflexión. Si hubiéramos decidido eliminar sólo la interrupción de teclado, hubieran sido necesarias las instrucciones MOV AL, 02h y OUT 21h, AL. Para volver a permitirla basta con ejecutar después MOV AL, 00h y OUT 21h, AL.

Por lo que se refiere al método de cambio del vector, actuando directamente sobre la TVI, en la práctica éste presenta una pega cuando se ejecuta paso-a-paso, mediante algún depurador, el cambio del vector de algunas interrupciones como la 10h (BIOS de vídeo), la 16h (BIOS de teclado) o la omnipresente 21h. Al ser empleadas continuamente, de forma directa o indirecta, por el programa, cuando el vector está a medio cambiar suele producirse la "catástrofe". La mejor solución a este problema es emplear el método del programa EJEMPLO2.ASM, que se comentará en breve. En el caso de la interrupción 05h no se tiene tanto problema, basta con no pulsar la tecla <Impr Pant> en un momento crítico.

OTRA POSIBILIDAD: CAMBIO INDIRECTO

El programa EJEMPLO2.ASM muestra otra forma de conseguir el mismo efecto que el anterior. La diferencia consiste en que el programador no debe manipular directamente la TVI, sino que confía esta labor a la funciones 25h y 35h del DOS (de la interrupción 21h).

La función 35h se emplea para obtener el contenido de un vector de interrupción. Depositando en AH el valor 35h y en AL el número de la interrupción que nos interesa, esta función devolverá en los registros ES y BX, el segmento y el desplazamiento del código gestor en cuestión.

La función 25h sirve para asignar a un fragmento de código un vector de interrupción. Requiere que, además de depositarse 25h en AH, y en AL el número de la interrupción, almacenemos en los registros DS y DX respectivamente, el segmento y el desplazamiento del código que se va a encargar de esa interrupción. En nuestro caso sólo se mueve el valor para el registro DX, ya que la única instrucción que va a componer el gestor se encuentra en el segmento de datos (recuérdese que en un archivo COM los cuatro segmentos están superpuestos).

Una vez ejecutado el programa en sí, se debe restaurar el vector de la int 05h. Por lo tanto, se recuperan los valores originales y se almacenan en DX y DS, por este orden. Esto tiene su importancia, ya que de no hacerse así se corre el gran riesgo de cometer un error habitual de principiante: si se carga el segmento en DS antes que el desplazamiento en DX, el operando de memoria que contiene el último se direccionará con respecto a una base que no es la original. Esto significa que el desplazamiento que se deposita en el vector va a ser falso con toda probabilidad, por lo que cuando se solicite esa interrupción se producirá la catástrofe. El problema se puede obviar empleando algún prefijo de segmento válido, pero es más cómodo respetar el orden arriba indicado.

CAMBIOS PERMANENTES: RUTINAS RESIDENTES

Con los programas anteriores se ha mostrado la forma de eliminar una interrupción durante la ejecución de un programa. Supongamos ahora que se desea dejar esa interrupción inhabilitada, o modificada permanentemente. Como es impensable que cada programa que se ejecuta incorpore el código necesario para este fin, la solución pasa por dejar residente en memoria este código hasta que el usuario lo considere oportuno.

El estudio riguroso de los programas residentes excede los objetivos de este capítulo. Sin embargo, en los programas IMPRESO.ASM, VIDEO.ASM y en sus correspondientes desinstaladores DESIMP.ASM y DESVIDEO.ASM, se muestra una aproximación a cómo dejar un conjunto de datos e instrucciones permanentemente en memoria, después de haber cambiado un vector de interrupción, y cómo liberar esa memoria restaurando dicho vector.

IMPRESO.ASM se encarga de inhabilitar la interrupción 5 de forma que el volcado de la pantalla por impresora no sea posible. Analizando su estructura se puede entender el método general que se debe seguir en cualquier caso similar.

El código del programa comienza con un salto incondicional a la etiqueta LanzaResidente. Las instrucciones posteriores a esta etiqueta van a realizar una serie de actividades, antes de dejar residente en memoria el fragmento de programa comprendido entre el desplazamiento 0000h y el dado por la última instrucción anterior a LanzaResidente.

En primer lugar se comprueba que ese fragmento no se encuentra ya activado. Para ello la subrutina BuscaID recorre la memoria hasta la dirección A0000h, buscando una señal que le permita decidir si el programa está residente o no. La prueba de que ya se efectuó el proceso de instalación, la obtendrá en caso de encontrar modificada en algún punto la secuencia de caracteres dada por la variable IdResidente. La función de esta variable es doble: por un lado informa con su presencia de que el proceso está activo y, por otro, sirve como punto de referencia para acceder a posiciones de memoria importantes en el proceso de desinstalación.

Si BuscaID encuentra la cadena "//INT05H\\", comprueba si los dos bytes que faltan almacenan los caracteres "##". Si es así, el proceso está ya en marcha y no es necesario volverlo a instalar. La rutina devuelve el valor 01h y el programa finaliza con la función DOS 4Ch. Si, por el contrario, al analizar los dos bytes que faltan, estos no son la secuencia "##", el procedimiento "deduce" que está chequeando la zona de memoria del propio programa instalador. En este caso se prosigue con la búsqueda hasta encontrar la secuencia buscada o llegar al límite permitido de la memoria. Este método de búsqueda es lento, pero es el que se empleaba antes de que la interrupción 2Fh fuera ampliada. Esta interrupción, cuyo uso requiere de una fase de preparación más compleja, se usará en el futuro al estudiar los programas residentes más a fondo.

EXCEPCIONES

Vector	Empleo
Ø0 h	División por cero
Ø1 h	Paso – A – Paso
Ø3 h	Punto de ruptura
Ø6 h	Código de operación no válido
Ø7 h	Coprocesador no presente

Figura 5.

Si el programa no estaba residente, se obtiene el valor del vector de la interrupción 05h y se almacena en las variables Offset05h y Segmento05h. Estas se emplearán para restaurar el vector cuando el programa se desinstale.

A continuación se sustituyen la cadena "00" por "##", señal para futuras ejecuciones de que ya se tiene el proceso en marcha, y después se hace que el vector 05h apunte a la instrucción IRET referenciada mediante la etiqueta InicioResidente. El siguiente paso es el de liberar mediante la función 49h parte de la memoria que el DOS reservó para el programa cuando éste fue cargado y que, en este caso, no va a ser de interés. Esto puede hacerse también en el proceso de desinstalación. La gestión de memoria dinámica bajo DOS se realiza con las funciones 48h, 49h y 4Ah.

Ahora sólo queda finalizar, dejando residente parte del programa que se cargó. De ello se ocupará la función 31h. Esta requiere tan sólo que se almacene en DX el número de párrafos deseados desde el inicio del segmento. Por este motivo se deja en DX el desplazamiento del primer punto del programa que no va a quedar residente y, dividiendo entre 16 (número de bytes de un párrafo) se obtiene la cantidad necesaria. Como la división pierde decimales, para estar seguros de que no se deja nada vital, se incrementa posteriormente en 1 DX. Acto seguido se ejecuta la interrupción 21h y el programa finaliza.

Otra posibilidad para dejar residente un fragmento de código, la tenemos en el uso de la interrupción 27h, presente ya en la primera versión del DOS. Está en desuso porque presenta inconvenientes que no plantea la función 31h, como la limitación a 64 Kb del tamaño de los residentes y la necesidad, a menudo incómoda, de que el registro CS apunte al inicio del PSP (prefijo de segmento de programa). La bibliografía citada al final del capítulo sirve para dar más información al respecto.

DESCARGA DE RUTINAS DE MEMORIA

Para eliminar de la memoria del sistema el programa anterior se suministra DESIMP.ASM. Este programa se limita a seguir una estrategia similar al anterior para comprobar si IMPRESO.COM ya había sido ejecutado. Si es así se anula la cadena poniendo "00" al comienzo de ésta y, mediante cálculos, se obtiene el segmento de inicio del fragmento residente. Esta información podría haberse guardado en una variable, como Offset05h y Segmento05h, pero es interesante comprobar cómo, dada una dirección física, se puede transformar en otra de formato más apropiado.

Ahora, usando este dato y teniendo localizado el valor del antiguo vector de interrupción, se restaura este último y se libera la memoria ocupada por el programa residente. Para liberar la memoria basta con apuntar con ES a la dirección del PSP activo cuando se ejecutó IMPRESO y emplear la función 49h del DOS.

INTERRUPCIONES SOFTWARE

	Vector	Empleo
BIOS	1Ø h	Driver de vídeo
	11 h	Configuración
	12 h	Tamaño memoria convencional
	13 h	Driver de disco
	14 h	Comunicaciones
	15 h	Extensiones 1/0
	16 h	Driver de teclado
	17 h	Driver de impresora
	19 h	Reinicialización
	1B h	Ctrl – Break – BIOS
DOS	2Ø h	Fin de programa
	21 h	Distribuidor funciones DOS
	22 h	Dirección de terminación
	23 h	Ctrl – Break DOS
	24 h	Tratamiento de errores
	2F h	Interrupción multiplex

Figura 6.

PONER Y QUITAR FUNCIONES

Otra posibilidad, más interesante que la anterior, es la de añadir, suprimir y/o modificar funciones de una interrupción, con el fin de aumentar sus prestaciones o evitar efectos indeseables en algunas ejecuciones.

Dado que ahora no se trata de eliminar una interrupción en bloque, el código residente no va a constar de una única y simple instrucción IRET. En su lugar tendremos un tratamiento de las distintas situaciones de interés que desembocará bien en un retorno de interrupción (para funciones nuevas o anuladas), bien en una transferencia de control a la verdadera interrupción mediante un salto condicional indirecto.

Cuando se trata de anular una función determinada, el caso más simple, el código del programa residente preguntará por el valor de AH. Si éste es el que se desea ignorar, el programa bifurcará a la instrucción IRET. En el programa ejemplo VIDEO.ASM se deja residente un fragmento que "parchea" la interrupción de vídeo (10h) impidiendo que se pueda escoger a través de ella un modo texto de resolución 40X25 (opciones 00h y 01h de la función 00h). Ejecutando el comando MODE después de la instalación de esta rutina se podrá comprobar su efectividad.

Modificar una función existente consiste en, una vez detectada, insertar instrucciones que produzcan los efectos deseados, antes de dejar que el control sea transferido al gestor real de la interrupción. Como se dijo antes, esto se consigue mediante un salto incondicional, para lo cual asignamos a las variables Offset10h y Segmento10h la etiqueta VectorInt10h. Ésta no es más que una referencia, que indica al procesador, que la información almacenada en las variables anteriores puede ser empleada en una llamada FAR indirecta, como una dirección física guardada como OFFSET, SEGMENTO. De esta forma la instrucción JMP CS:VECTORINT10H se traduce del modo adecuado. Obsérvese la presencia del prefijo CS: Dado que DS, base para este tipo de operandos, tiene un valor no admisible cuando se ejecuta el fragmento, es necesario referenciar las posiciones de memoria con el único segmento del que podemos estar seguros. Es muy habitual en estas circunstancias olvidarse de este detalle, con los efectos que se puede suponer. El programa VIDEO.ASM modifica el funcionamiento de las funciones 06h y 07h, forzándolas a borrar la pantalla con el atributo 1Eh. De esta manera es imposible, en modo texto y a través de esta interrupción, tener otros colores en pantalla.

En cuanto al aspecto de añadir funciones nuevas, puede verse como uno de los dos casos anteriores. La única salvedad es, que el número de función por el que se pregunta no existe para la interrupción dada. El programador le asigna el código que considera oportuno y, tras su ejecución, se retorna con IRET, o se transfiere el control a la interrupción real.

Evidentemente, para aquellas funciones no modificadas, el fragmento residente ha de ser invisible, y todas han de ser ejecutadas como siempre por el gestor original.

UN CASO ESPECIAL

En algunas ocasiones, cuando se manipula una interrupción que ofrece muchos servicios (la 21h, por ejemplo), es necesario recurrir a alguno de ellos desde el fragmento residente que la precede. La solución más apropiada consiste en invocar el gestor principal mediante una llamada CALL indirecta, utilizando el contenido de las variables Offset y Segmento adecuadas. La llamada funcionará de cualquier manera, pero para que el retorno sea posible es necesario tener en cuenta que la instrucción INT almacena en la pila los flags actuales y el segmento y el desplazamiento de la instrucción siguiente. Como CALL sólo almacena los dos últimos, la instrucción IRET con la que va a finalizar el gestor, sacará de la pila más de lo que se introdujo: para solucionar este problema se debe ejecutar la instrucción PUSHF (flags a la pila) antes de la instrucción CALL.

POR ÚLTIMO

Visto lo anterior, la creación de nuevas interrupciones, carece de misterio. El proceso que se debe seguir es el mismo que el empleado para instalar código residente. La única salvedad es que el vector de interrupción que se utiliza no está asignado a ningún gestor antes de instalarse el proceso.

En teoría existe un gran cantidad de entradas no empleadas en la TVI. Sin embargo, sólo se tendrá la seguridad de no estar empleando un vector ocupado al emplear los correspondientes a los números comprendidos entre el 60h y el 66h, reservados por convenio a las interrupciones de usuario.

Codificar e instalar una nueva interrupción tiene su máxima utilidad cuando se está trabajando en una aplicación con varios programas con requerimientos comunes, no satisfechos por las interrupciones estándar. La solución óptima es, en este caso, la instalación de una rutina de servicios que no tenga la necesidad de estar presente en cada uno de los programas que la necesitan. Esta es una ventaja de las interrupciones sobre los procedimientos: su omnipresencia. Las desventajas, en cambio, estriban en que el proceso de llamada es más lento, al tratarse de llamadas siempre indirectas y con un mayor número de pasos internos. ■

BIBLIOGRAFÍA

Ray Duncan, La ROM BIOS de IBM, EDICIONES ANAYA MULTIMEDIA, Madrid, 1989

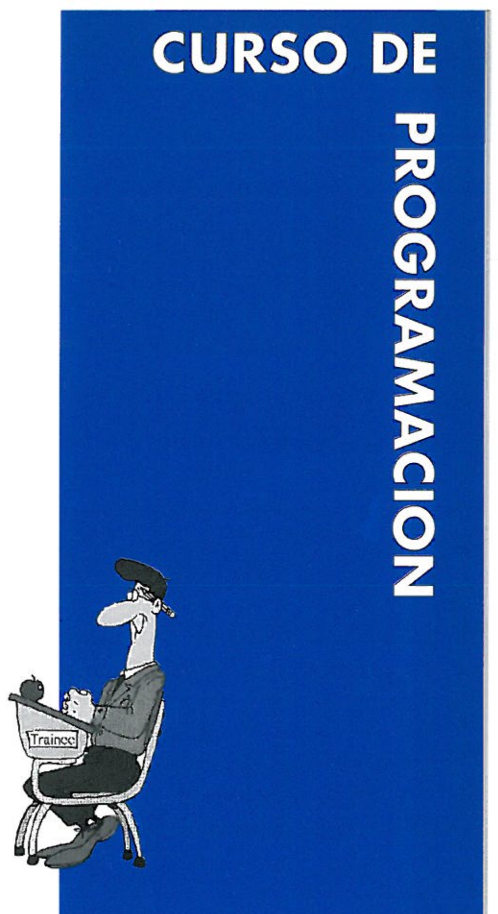
Ray Duncan, Funciones del MS-DOS, EDICIONES ANAYA MULTIMEDIA, Madrid, 1989

Robert Jourdain, Solucionario del programador para IBM PC, XT, AT y compatibles, EDICIONES ANAYA MULTIMEDIA, Madrid, 1988

Angulo-Funke, 386 y 486 Microprocesadores avanzados de 32 bits, De. Paraninfo, Madrid, 1992.

COMPONENTES ELEMENTALES

Miguel Angel Alcalde



El ordenador es sin duda la herramienta más versátil que el hombre haya construido nunca. Se puede emplear como máquina de escribir, como calculadora o incluso para conducir una carretilla por un almacén robotizado. Pero cada acción, por pequeña que sea debe ser programada por una persona.

El lenguaje empleado posee generalmente una serie de ordenes básicas como leer o escribir predefinidas, es decir, el ordenador sabe como se lee por ejemplo una letra del teclado o cómo se escribe en el monitor (En Pascal READ, WRITE.

El lenguaje C no posee ninguna estando definidas a través de librerías). Además también tiene una forma concreta de decir las cosas y de dirigir la ejecución de cada instrucción, que es la característica principal que diferencia los distintos lenguajes (Pascal C, ADA, BASIC, ...).

ALMACENAMIENTO DE DATOS

La memoria de un computador puede representarse como una gran caja dividida en celdillas capaces de contener únicamente un 1 o un 0. Todas las celdas están numeradas siendo posible el acceso directo a cualquiera de ellas a través de su referencia.

Los lenguajes de alto nivel permiten manejar y organizar esta estructura de la forma más conveniente al programador.

Esto se realiza mediante las variables, que son un conjunto de celdillas a las que se les pone un nombre para referenciarlas donde el programador puede colocar y modificar cualquier dato siempre que corresponda con la forma física de las casillas.

Para poder ser utilizadas es necesario declararlas previamente en casi todos los lenguajes convencionales, en BASIC no se requiere.

Los tipos más comunes de variables son los siguientes:

- LÓGICO: Sólo existe en Pascal. Es la más sencilla, posee una sola celda capaz de almacenar un 1 o un 0, aunque la implementación real es oculta. Se emplea para contener valores lógicos (TRUE / FALSE).

Declaración:

```
Pascal:         variable: Boolean;
```

Asignación:

```
Pascal:  variable:=TRUE;
C: .0 falso y mayor que 0 verdad.
QBASIC:  0 falso y -1 verdad.
```

- CARÁCTER: Contiene un carácter o símbolo perteneciente a la tabla ASCII. Contiene 8 posiciones necesarias para codificar los 256 caracteres.

La memoria de un computador puede representarse como una gran caja dividida en celdillas

Declaración:

```
Pascal:         variable: Char;
C:              char variable;
```

Asignación:

```
Pascal:         variable:='a';
C:              variable='a';
QBASIC:         variable$='a'
```

En el anterior capítulo se describió, a grandes rasgos, cómo es un ordenador y qué expresiones había que emplear para comunicarle las ordenes que debe ejecutar. En este artículo se describirán las sentencias de control más comunes en los lenguajes de programación así como la forma de almacenar la información en la memoria del ordenador.

- NÚMERO ENTERO: Permite manejar números enteros desde el -32768 al 32768. Emplea 16 casillas o posiciones de memoria. Se suelen emplear como contadores dentro de los ciclos. Declaración:

Pascal:	variable: Integer;
C:	int variable;

Asignación:

Pascal:	variable:=10;
C:	variable=10;
QBASIC:	variable=10

- NUMERO REAL: Son números de tipo REAL, el rango y la precisión dependen del ordenador. En general suelen estar codificados por 32 bits y pueden contener números entre el 3.4E-38 al 3.4E+38. Se emplean para efectuar cálculos matemáticos.

Declaración:

Pascal:	variable: real;
C:	float variable;

Asignación:

Pascal:	variable:=3.14;
C:	variable=3.14;
QBASIC:	variable%=3.14

EXPRESIONES

En el punto anterior se ha visto que se puede asignar un valor a una variable, pero también es posible darle el valor resultante tras analizar una expresión como por ejemplo 2+3. En general una expresión es un conjunto de operaciones de cualquier tipo que como resultado se obtiene un valor de un tipo concreto (carácter, número, lógico, ...).

Para poder construir las expresiones es necesario conocer que operadores existen y de que tipo son:

- Lógicos: Y, O, NO, Y/O. Se emplean en las condiciones de las sentencias de control. Se obtiene un valor lógico.

Pascal:	AND, OR, NOT, XOR.
C:	&&, ||, !.
QBASIC:	AND, OR, NOT.

- Relaccionales: Menor, mayor, igual, menor o igual, mayor o igual, distinto. Se obtiene un valor lógico y se emplean en las condiciones de las sentencias de control.

Pascal, QBASIC:	<, >, =, <=, >=, <>.
C:	<, >, ==, <=, >=, !=.

- Matemáticos: Suma, resta, multiplicación, división entera (sólo en Pascal), resto de la división entera, división real.

Pascal:	+, -, *, DIV, MOD, /.
C:	+, -, *, (no), %, /.
QBASIC:	+, -, *, (no), MOD, /.

Para construir estas expresiones se puede utilizar los paréntesis para especificar como se evalúan. Por ejemplo:

- (1+2)*2 es distinto que 1+(2*2)
- (1+2)>=3 se obtiene el valor lógico VERDAD.

También se puede jugar con el orden de precedencia de cada operador: La multiplicación, división y resto se evalúan antes que la suma o la resta. En los operadores lógicos y relaccionales, la negación es la primera, le siguen las comparaciones de menor o mayor, después la igualdad y distinto, y por último Y y O. Por ejemplo:

- 1 Y NO 0 O 1 da como resultado VERDAD.
- 1 Y NO (0 = 1) da como resultado FALSO.

La sentencia EN CASO DE sirve para simplificar la sentencia SI

SENTENCIAS DE CONTROL

Las sentencias de control permiten especificar cuantas veces y de que forma se debe repetir una serie de acciones o si se debe o no realizar instrucciones dependiendo de unas condiciones.

Las más importantes son las expuestas a continuación:

SI

Sintaxis: SI condición ENTONCES acciones1 EN OTRO CASO acciones2. Es posible anidar sentencias SI unas dentro de otras. La parte de EN OTRO CASO puede ser omitida, quedando la expresión: SI condición ENTONCES acciones;.

```
Pascal:  IF condición THEN acciones1
         ELSE acciones2;
C:       IF (condición) acciones1
         ELSE acciones2;
QBASIC:  IF condición THEN acciones1
         ELSE acciones2
         END IF
```

Significado: Si el resultado de la condición es verdad (en C distinto de 0) se ejecutan las "acciones1", si es mentira o 0 si es C, se ejecutan las "acciones2".

```
Seudocódigo:
         si 1=1 entonces escribir("lógico, ...")
         en otro caso escribir("¡Es imposible!")

Pascal:  If 1=1 then write("Lógico, ...")
         else write("¡Es imposible!");
C:       if (1==1) printf("lógico, ...")
         else printf("¡Es imposible!");
QBASIC:  IF 1=1 THEN print "Lógico, ..."
         ELSE print "¡Es imposible!"
         END IF
```

EN CASO DE

Sintaxis: EN CASO DE variable valor1: acciones1; valor2: acciones2; ... valorN: accionesN; FIN CASO;. En algunos lenguajes como C, BASIC, puede emplearse EN OTRO CASO como último valor: EN CASO DE variable valor1: accion1; ... EN OTRO CASO accionesX; FIN CASO;.

```
Pascal:  Case variable of
         valor1: acciones1;
         ...
         valorN: accionesN;
         ELSE accionesX;(Sólo en T. Pascal)
         End;
C:       Switch (variable)
         {
         case valor1: acciones1; break;
         case valor2: acciones2; break;
         ...
         case valorN: accionesN;
         default: accionesX;
         }
QBASIC:  SELECT CASE variable
         CASE valor1: acciones1
         CASE valor2: acciones2
         ...
         CASE ELSE accionesX
         END SELECT
```



Significado: Sirve para simplificar la sentencia SI cuando han de compararse distintos valores para una misma variable. Ejemplo:

```
Si variable=1 entonces acciones1
else si variable=2 entonces acciones2
else si variable=3 entonces acciones3
else ...

En caso de variable
        1: acciones1;
        2: acciones2;
        3: acciones3;
        ...
Fin caso;
```

Es evidente que codificar la sentencia CASE es más cómodo que la SI, en este caso.

PARA

Sintaxis: PARA variable=valor1 HASTA valor2 HACER acciones;

Las repeticiones que se van a efectuar son conocidas por el programador. En algunos lenguajes se puede modificar el incremento de la variable que indica el número de repeticiones. El contador variable no debe ser modificado dentro del cuerpo del ciclo, aunque puede ser consultado.

```
Pascal:
For variable:=valor1 to valor2 DO acciones; o
For variable:=valor2 downto valor1 acciones;
C:
 For(variable=valor1; variable comparada con
valor2; incremento o decremento de la variable)
        acciones;
QBASIC:
        FOR variable=valor1 TO valor2
        STEP valorIncremento
                acciones
        NEXT variable
```

Significado: Repite acciones un número determinado de veces. Sirve para recorrer matrices o para realizar cálculos. Por ejemplo:

```
Pascal: For i:=1 to 10 do
        begin
                acción1;
                ...
        end;
C:      for (i=1; i<=10 ; i++)
        {
                acción1;
                ...
        }
QBASIC  FOR i=1 TO 10 STEP 1
                acción1
                ...
        NEXT i
```

MIENTRAS

Sintaxis: MIENTRAS condición HACER acciones;. En cada repetición se comprueba antes de comenzar si se cumple la condición, si es así se ejecutan las instrucciones del cuerpo del ciclo, en otro caso se salta hasta la línea siguiente después de esta estructura.

Pascal:

While condición do acciones;

C: while (condición) acciones;

QBASIC:

WHILE condición acciones WEND

Significado: Repite un número indeterminado de veces las instrucciones indicadas. Se emplea en recorrido de ficheros o listas. La condición de terminación puede ser modificada durante la ejecución del ciclo para forzar su terminación. Por ejemplo:

```
Pascal: While cierto do
        begin
                acción1;
                acción2;
        .       ..
        end;
C:      While (cierto)
        {
                acción1;
                acción2;
        .       ..
        }
QBASIC  WHILE cierto
                acción1
                acción2
        .       ..
        WEND
```

REPETIR

Sintaxis: REPETIR acciones HASTA condición;. Es similar al MIENTRAS excepto que la comprobación para volver a ejecutar el ciclo se realiza tras haberse ejecutado las instrucciones. Hay alguna pequeña diferencia entre las implementaciones en Pascal y C, ya que en el primer caso la comprobación se hace hasta que se cumple la condición, mientras que en el segundo se repite mientras se cumpla la condición. QBASIC acepta ambas formas.

```
Pascal: Do acciones until condición;
C: do acciones while condición;
QBASIC:
        DO acciones LOOP UNTIL condición
        o
        DO acciones LOOP WHILE condición
```

Significado: Repite una serie de instrucciones mientras / hasta que se

SEUDOCÓDIGO

Notas aclaratorias:

[v] es el nombre de una variable.

[M N] son valores o constantes de tipo número.

[;] es el separador de sentencias.

[,] es el separador de elementos.

[.] es el indicador de fin de programa.

[+ - * / =] son operadores matemáticos.

[no si menor mayor igual o y] son operadores lógicos.

```
nombre : tipo;

repetir
...
hasta condición;

desde v=N hasta M sentencia;
o
desde v=N hasta M
...
fin desde;

mientras condición sentencia;
o
mientras condición
...
fin mientras;

si condición entonces sentencia;
o
si condición entonces
...
fin si;
o
si condición entonces
...
en otro caso
...
fin si;
```

Tipos de datos:

Una variable pude contener un número ENTERO, REAL, o un CARACTER o ...

Sentencia:

Es una orden o conjunto de ordenes. Por ejemplo, una asignación, o un procedimiento.

Asignación:

```
v=N;
o
v=expresión;
```

Expresión:

Es una composición de operaciones entre distintos datos del mismo tipo o compatibles. Por ejemplo:

```
((Verdad y (falso o Verdad))
2+3
PI / raíz2(20) + e
'ésto en un tro'+'zo que continua'
((1 menor 2) y verdad)
```

Condición:

Es una expresión que tras evaluarla devuelve un resultado lógico. Por ejemplo:

```
((2+2) igual 4)
(1 menor 2)
(no (verdad) y falso)
```

Declaración de subprogramas:

Tipos de parámetros:

E/S: Permite modificar la variable recibida.

E: Sólo se puede leer el contenido.

Sintaxis de los parámetros:

[tipo del parámetro] [nombre de la variable] : [tipo]

Si hay más de uno se separan por [;]

Ejemplos prácticos:

```
procedimiento Escribe (E/S número: entero;
                       E palabra: CadenaCaracteres);
 [declaración de variables]
 [sentencias]
fin Escribe;

función Suma1 (E número: entero) : entero;
 [declaración de variables]
 [sentencias]
 Suma1= número + 1;
fin Suma1;
```

Estructura de un programa principal:

```
[declaración de constantes]
[declaración de tipos nuevos]
[declaración de variables globales]
[declaración de procedimientos y funciones]

programa [nombre del programa]
 [sentencias]
fin [nombre del programa]
```

cumpla la condición. Para hacer equivaler MIENTRAS condición a HASTA condición, basta con negar la condición del MIENTRAS. Por ejemplo:

```
Pascal:
    Do
            acción1;
            ...
    until condición;
C:
    do
    {
            acción1;
            ...
    }
    while (not condición);
QBASIC:
    DO
            acción1
            ...
    LOOP UNTIL condición
    o
    DO
            acción1;
            acción2;
            ...
    LOOP WHILE not condición
```

EL PRÓXIMO CAPÍTULO

Tras esta concreción de los elementos básicos para programar, es necesario pasar directamente a programar.

A partir del siguiente artículo se utilizarán el seudocódigo *SóloP* y Pascal como lenguaje de programación real, por lo que se recomienda adquirir un compilador de Pascal para seguir con aprovechamiento este curso de programación básica. ■

BIBLIOGRAFÍA

Turbo C/C++, Manual de referencia.
Autor: Herbert Schildt.
Editorial: McGraw-Hill
Turbo Pascal, Manual de referencia.
Autor: Herbert Schildt.
Editorial: McGraw-Hill.
Programación en QBASIC.
Autor: Luis Joyanes.
Editorial: McGraw-Hill.

CORREO LECTORES

Para cualquier duda referente a los temas/árticulos tratatos en la revista, escriba una carta a:

Sólo Programadores
Referencia: *Correo Lectores*
TOWER COMMUNICATIONS
C/ Marqués de Portugalete, 10 Bajo
Madrid 28027

COMO SUSCRIBIRSE A



ıscríbase enviando este cupón por correo o fax (91) 320.60.72, o llamando al teléfono (91) 741.26.62 Horario 9 a 14 y 15:30 a 18:30 h.

eseo suscribirme a la revista SÓLO PROGRAMADORES acogiéndome a la siguiente modalidad:

☐ Suscripción: 1 año + regalo (filtro monitor) **por sólo** 10.995 ptas. Estudiantes carreras técnicas 40% Dto.: 8.250 ptas.

lombre y apellidos...Domicilio..

ɔblación...C.P ....Provincia.........................Telf..............................Profesión...................

ORMA DE PAGO:

❏ Con cargo a mi tarjeta VISA nº ....

Fecha de caducidad de la tarjeta................Nombre del titular, si es distinto..

❏ Domiciliación bancaria.

Señor Director del banco ....

Población ....

CODIGO CUENTA CLIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

Ruego a vd. que se sirva cargar en mi ☐ cuenta corriente ☐ libreta de ahorro número...

el recibo que le será presentado por TOWER COMMUNICATIONS, S.R.L. como pago de mi suscripción a la revista SÓLO PROGRAMADORES.

❏ Contra-reembolso del importe más gastos de envío.

❏ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.

❏ Giro Postal (adjunto fotocopia del resguardo).

Firma:

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Marques de Portugalete 10, Bajo
28027 Madrid.

USO DE TABLAS DESLIZANTES

Juan Manuel y Luis Martín

Desde la incorporación de la programación orientada a objetos (OOP) a las últimas versiones de Clipper, varias de sus herramientas han alterado su funcionamiento, dotando así al programador de un mayor control sobre los distintos procesos.

Una de las herramientas que más mejoras ha incorporado es la visualización de la información almacenada en bases de datos mediante el uso de tablas deslizantes.

Una de las mayores novedades de las últimas versiones de Clipper consiste, en la incorporación al lenguaje de algunas técnicas de programación orientada a objetos. A pesar de ello, Clipper no puede ser considerado un lenguaje puro orientado a objetos, ya que sólo es posible utilizar cuatro clases predefinidas. Estas cuatro clases son las siguientes:

- Get: proporciona objetos para editar campos y variables
- TBrowse: proporciona objetos para realizar tablas deslizantes
- TBColumn: proporciona objetos para definir las columnas de las tablas deslizantes
- Error: proporciona objetos con información sobre errores

Al igual que otros lenguajes, los objetos en Clipper se crean a partir de una clase, y están compuestos de atributos y métodos. Para la creación de un objeto a partir de una clase, se debe invocar el método constructor de dicha clase. El resultado de ésta operación deberá ser asignado a una variable objeto. El acceso tanto a los métodos como a los atributos de un objeto se realiza mediante el envío de mensajes. Para ello se utiliza el operador ":", cuya sintaxis es la siguiente:

```
Objeto:Método( [ListaArgumentos] )
Objeto:Atributo
```

TABLAS DESLIZANTES

Una de las técnicas más habituales para la presentación en pantalla de la información almacenada en bases de datos, es la utilización de tablas deslizantes. Se trata de tablas en forma de ventana, en las que se visualiza el contenido de los distintos campos y registros de las bases de datos en forma de filas y columnas. Este tipo de estructuras permite al usuario desplazarse por ellas, seleccionar información, editarla, etc. La figura 1 muestra el aspecto de una tabla deslizante que visualiza el contenido de una base de datos.

Hasta la versión 5.00 del lenguaje Clipper, esta operación se venía realizando con la función DBEDIT(). Sin embargo, a partir de dicha versión, es posible realizarla mediante objetos derivados de las clases TBrowse y TBColumn.

El objeto creado a partir de la clase TBrowse representa la propia tabla. La clase TBColumn es una subclase de la anterior, de forma que, los objetos creados a partir de ésta subclase representan la definición de las distintas columnas que constituyen la tabla.

CREACIÓN DE UNA TABLA DESLIZANTE

El primer paso para la creación de una tabla deslizante es la creación de un objeto de la clase TBrowse. Para ello, se utilizará el método constructor de la clase, asignando su resultado a una variable de objeto. Aunque existe un método constructor general, TBrowseNew(), cuando la tabla deslizante va a ser utilizada para mostrar la información almacenada en una base de datos, se utiliza un constructor más especializado, cuya sintaxis es la siguiente:

```
oTabla := TBrowseDB( nSup, nIzq, nInf, nDer )
```

Este constructor crea el mismo tipo de objeto que el constructor general de la clase. La única diferencia existente entre los objetos creados con uno u otro constructor es que TBrowseDB() asigna valores directamente a los atributos de movimiento, no siendo necesario preocuparse por su contenido.

Los argumentos pasados en la llamada al constructor representan las coordenadas de la esquina superior izquierda y de la esquina inferior derecha del recuadro de la pantalla donde va a ser visualizada la tabla.

Una vez creado el objeto de la tabla deslizante, deben crearse tantos objetos de la clase TBColumn como columnas vaya a contener la tabla deslizante. Para crear un objeto de esta clase debe utilizarse el método constructor de la misma, asignándose el resultado a variables objeto:

```
oCol := TBColumnNew( cCabecera, bBloque )
```

En la invocación al método constructor de la clase TBColumn deben especificarse dos argumentos. Por un lado, se especifica una cadena de caracteres que constituirá la cabecera de la columna. Por otro, debe especificarse un bloque de código que será el encargado de devolver los distintos valores que se van a mostrar en dicha columna. Los valores que serán mostrados en la columna se corresponderán con los valores devueltos por el bloque de código, y pueden ser de cualquier tipo de datos.

Si, por ejemplo, se desea visualizar la información almacenada en la base de datos CLIENTES.DBF, cuya estructura se muestra en la tabla 1, será necesario crear un objeto columna por cada campo:

```
oTabla := TBrowseDB( 5, 10, 19, 69 )

oCol1 := TBColumnNew( "Número", { || clientes->numero } )
oCol2 := TBColumnNew( "Nombre", { || clientes->nombre } )
oCol3 := TBColumnNew( "Dirección", { || clientes->direccion } )
oCol4 := TBColumnNew( "Teléfono", { || clientes->telefono } )
oCol5 := TBColumnNew( "Fª Alta", { || clientes->fecalta } )
oCol6 := TBColumnNew( "Empresa", { || clientes->empresa } )
```

Una vez creados todos los objetos columna, éstos deben ser añadidos a la tabla deslizante. Para ello, se utiliza el método AddColumn() de la clase TBrowse, cuya sintaxis es la siguiente:

```
oBr:AddColumn( oCol )
```

Para el ejemplo anterior, referente a la base de datos CLIENTES.DBF, será necesario añadir las columnas creadas:

```
oTabla:AddColumn( oCol1 )
oTabla:AddColumn( oCol2 )
oTabla:AddColumn( oCol3 )
oTabla:AddColumn( oCol4 )
oTabla:AddColumn( oCol5 )
oTabla:AddColumn( oCol6 )
```

Es posible evitar el uso de variables de columna intermedias, añadiendo directamente al objeto tabla los objetos columna generados por el constructor de la clase TBColumn, de la forma siguiente:

```
oTabla:AddColumn( TBColumnNew( cCabecera, bBloque ) )
```

A medida que se van añadiendo las columnas a la tabla deslizante, éstas quedan numeradas internamente en el orden en que han sido introducidas. Este será el orden en que serán visualizadas en la pantalla.

TABLA 1

Estructura de CLIENTES.DBF

Nombre	Tipo	Longitud	Decimales
NUMERO	Numérico	5	0
NOMBRE	Caracteres	30	0
DIRECCION	Caracteres	30	0
TELEFONO	Caracteres	10	0
FECALTA	Fecha	8	0
EMPRESA	Lógico	1	0

PERSONALIZACIÓN DE LA TABLA Y LAS COLUMNAS

Es posible personalizar, tanto la tabla completa como cada una de las columnas que la componen, mediante la asignación de valores a los atributos de cada uno de los objetos. Las tablas 2 y 3 muestran una relación completa de los atributos de las clases TBrowse y TBColumn, respectivamente.

TABLA 2

Atributos de la clase TBrowse

Atributo	Descripción
AutoLite	Resalta o no el dato donde está el cursor
ColCount	Número de columnas de la tabla
ColorSpec	Cadena de color para la tabla
ColPos	Columna donde está el cursor
ColSep	Carácter separador de columnas
FootSep	Carácter separador de pies
Freeze	Número de columnas congeladas
GoBottomBlock	Bloque de código para ir al final
GoTopBlock	Bloque de código para ir al principio
HeadSep	Carácter separador de cabeceras
HitBottom	Indica si está al final de la tabla
HitTop	Indica si está al principio de la tabla
LeftVisible	Columna más a la izquierda no congelada
nBottom	Coordenada inferior de la tabla en pantalla
nLeft	Coordenada izquierda de la tabla en pantalla
nRight	Coordenada derecha de la tabla en pantalla
nTop	Coordenada superior de la tabla en pantalla
RightVisible	Columna más a la derecha visualizada no congelada
RowCount	Número de filas visibles en la tabla
RowPos	Fila donde está el cursor
SkipBlock	Bloque de código para posicionar el puntero
Stable	Indica si la tabla está estable

Para el correcto funcionamiento de la tabla deslizante, existen tres atributos que definen el movimiento del cursor a través de la misma. Los atributos GoTopBlock y GoBottomBlock deben contener bloques de código para posicionar el puntero en el primer y último registro, respectivamente. Igualmente, el atributo SkipBlock debe contener un bloque de código para posicionar el puntero, saltando tantas posiciones como se le indique mediante un argumento. Si se está visualizando una base de datos, y se ha utilizado el constructor TBrowseDB(), no será necesario especificar estos atributos, ya que por defecto se inicializan con las operaciones GO TOP, GO BOTTOM y SKIP, respectivamente.

Toda la tabla, incluidas las columnas, utilizan una misma cadena de color que se encuentra almacenada en el atributo ColorSpec de la clase TBrowse. Por defecto, esta cadena se corresponderá con la cadena indicada por la función SETCOLOR().Las cabeceras, los separadores, los pies y los valores no seleccionados se visualizan en el color del primer ámbito. Los valores seleccionados, es decir, aquellos sobre los que se encuentra el cursor, se visualizan en el color del segundo ámbito.

La selección del color de cada una de las columnas puede alterarse especificando dos ámbitos distintos mediante el atributo DefColor de la clase TBColumn. En dicho atributo, los ámbitos se indican mediante una matriz de dos números que, por defecto, será {1,2}. Estos números se corresponden con cada uno de los cinco ámbitos de la cadena ColorSpec.

También es posible indicar los ámbitos de color en función del valor de cada uno de los elementos de la columna, utilizando para ello un bloque de código que se almacena en el atributo ColorBlock de la clase TBColumn.

Cuando no es posible visualizar todas las columnas a la vez, la tabla deslizante se encarga de hacerlas aparecer y desaparecer, dependiendo del movimiento del cursor. Sin embargo, es posible dejar fijas a la izquierda un número de columnas, denominadas columnas congeladas. Este número se indica en el atributo Freeze de la clase TBrowse.

TABLA 3

Atributos de la clase TBColumn

Atributo	Descripción
Block	Bloque de código para la obtención de datos
Cargo	Variable de propósito general
ColorBlock	Bloque de código para el color de los valores
ColSep	Separador izquierdo de la columna
DefColor	Matriz para definir el color
Footing	Literal del pie de la columna
FootSep	Carácter separador del pie
Heading	Literal de la cabecera de la columna
HeadSep	Carácter separador de la cabecera
Width	Ancho de visualización de la columna

EL PROCESO DE VISUALIZACIÓN

Una vez especificadas las características propias de la tabla deslizante, ésta debe ser visualizada en la pantalla. El manejo de una tabla deslizante consiste en el procesamiento de las teclas que el usuario va pulsando, y la visualización en pantalla de la información correspondiente. Las operaciones necesarias para la obtención de los valores de las distintas filas y columnas, así como la visualización de los mismos, consumen un cierto tiempo. Este proceso se conoce como estabilización del objeto. Hasta que no ha finalizado el proceso de estabilización del objeto, no se garantiza que la información visualizada sea la correcta. Si, por ejemplo, durante la visualización de una base de datos se pulsa la tecla [AvPág], la tabla deslizante debe calcular los valores de todas las columnas para el nuevo conjunto de registros visualizado.

El control de este proceso puede realizarse de dos formas distintas. La más sencilla, consiste en la utilización del método ForceStable(). Este método se encarga de realizar la estabilización completa del objeto, sin posibilidad de que ésta sea interrumpida. De esta forma, cuando finaliza la ejecución de éste método, el objeto se encuentra estable, pudiendo entonces pasar a procesarse la siguiente tecla.

```
WHILE .T.
  oTabla:ForceStable()
  nTecla := INKEY( 0 )
  ProcesoTecla( nTecla )
ENDDO
```

Para procesar las teclas, puede utilizarse una estructura CASE que realice distintas operaciones dependiendo de la tecla que haya sido pulsada. Entre estas operaciones pueden incluirse llamadas a los métodos de movimiento del cursor de la tabla deslizante, mostrados en la tabla 4. También pueden incluirse otras funciones de usuario que ayuden a personalizar el funcionamiento de la tabla deslizante.

TABLA 4

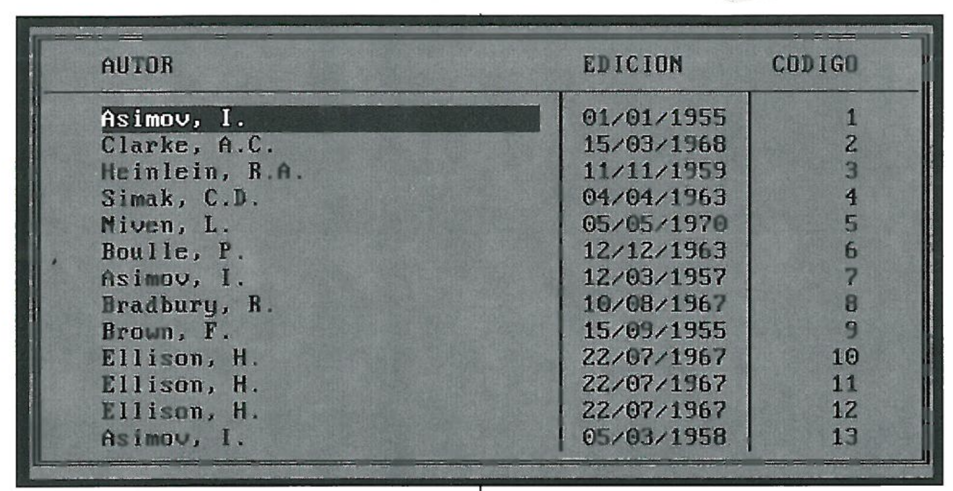
Métodos de la clase TBrowse

Método	Descripción
AddColumn()	Añade una columna a la tabla
ColorRect()	Define una zona que se visualiza en otro color
ColWidth()	Devuelve el ancho de visualización de una columna
Configure()	Carga los valores por defecto de los atributos
DeHilite()	Elimina el resaltado del cursor
DelColumn()	Elimina una columna de la tabla
ForceStable()	Fuerza la estabilización de la tabla
GetColumn()	Devuelve un objeto columna
Hilite()	Resalta la línea donde está el cursor
InsColumn()	Inserta una columna en una posición de la tabla
Invalidate()	Redibuja toda la tabla tras estabilizarse
RefreshAll()	Redibuja los datos de la tabla tras estabilizarse
RefreshCurrent()	Redibujar sólo la fila actual tras estabilizarse
SetColumn()	Reemplaza una columna por otra
Stabilize()	Estabiliza la tabla paso a paso
Down()	Mueve el cursor a la siguiente fila
End()	Mueve el cursor a la última fila de la pantalla
GoBottom()	Mueve el cursor a la última fila de la tabla
GoTop()	Mueve el cursor a la primera fila de la tabla
Home()	Mueve el cursor a la primer columna de la pantalla
Left()	Mueve el cursor a la columna anterior
PageDown()	Mueve el cursor una página más abajo
PageUp()	Mueve el cursor una página más arriba
PanEnd()	Mueve al cursor a la última fila de la tabla
PanHome()	Mueve el cursor a la primera fila de la tabla
PanLeft()	Visualiza las primeras filas de la tabla
PanRight()	Visualiza las últimas filas de la tabla
Right()	Mueve el cursor a la siguiente columna
Up()	Mueve el cursor a la fila anterior

ESTABILIZACIÓN POR ETAPAS

Como se ha mencionado anteriormente, el proceso de estabilización puede consumir bastante tiempo, cuando se procesan grandes cantidades de información. Aunque es conveniente no interrumpir el proceso de estabilización hasta que éste haya sido





AUTOR	EDICION	CODIGO
Asimov, I.	01/01/1955	1
Clarke, A.C.	15/03/1968	2
Heinlein, R.A.	11/11/1959	3
Simak, C.D.	04/04/1963	4
Niven, L.	05/05/1970	5
Boulle, P.	12/12/1963	6
Asimov, I.	12/03/1957	7
Bradbury, R.	10/08/1967	8
Brown, F.	15/09/1955	9
Ellison, H.	22/07/1967	10
Ellison, H.	22/07/1967	11
Ellison, H.	22/07/1967	12
Asimov, I.	05/03/1958	13

FIGURA 1: Visualización del contenido de una base de datos mediante una tabla deslizante.

completado, existen ciertas ocasiones en que interesa realizar determinadas operaciones aunque el objeto no se encuentre estable. Un ejemplo típico es el caso en que el usuario desea abandonar la visualización en cualquier momento. Para estas ocasiones, la clase TBrowse proporciona un atributo y un método que permiten realizar la estabilización del objeto por etapas.

```
FUNCTION ProcesoTecla( nTecla )
 DO CASE
  CASE nTecla == K_UP
   oTabla:Up()
  CASE nTecla == K_HOME
   oTabla:GoTop()
  ...
  CASE nTecla == K_ESC

   Terminar()
 ENDCASE
RETURN NIL
```

El atributo Stable contiene un valor de tipo lógico que indica si el objeto se encuentra estable o no. Por otro lado, el método Stabilize() permite realizar el proceso de estabilización del objeto paso a paso. Cuando se ejecuta este método por primera vez, se estabilizan las cabeceras de las columnas. En las siguientes ejecuciones se consigue la estabilización de cada una de las líneas de datos del objeto, estabilizándose una línea por cada ejecución. Por tanto, la ejecución del método ForceStable() puede simularse mediante el siguiente bucle:

Para interrumpir el proceso de estabilización del objeto, bastará con realizar alguna llamada en el interior del bucle. Si, además, se desea que dicha interrupción sea consecuencia de la pulsación de una tecla, puede utilizarse un bucle como el siguiente:

```
DispBegin()
WHILE !oTabla:Stable
 oTabla:Stabilize()
ENDDO
DispEnd()
```

CUADRO 1

```
#include "DBSTRUCT.CH"
#include "INKEY.CH"

FUNCTION Visor( cBase )

 LOCAL aDbf := {}, oDbf, cCampo, nTecla, cColor

 //Si no se pasa argumento, visualizar sintaxis
 IF cBase == NIL
  QOUT( "Sintaxis: VISOR [ruta]fichero.dbf" )
  RETURN NIL
 ENDIF
 //Si no se encuentra fichero, terminar
 IF !FILE( cBase )
  QOUT( "No se encuentra el fichero " + cBase )
  RETURN NIL
 ENDIF
 // Inicializar el entorno
 cColorAnt := SET( _SET_COLOR, "N/BG,W/B+" )
 SET( _SET_DATEFORMAT, "dd/mm/yyyy" )
 // Abrir la base de datos
 USE (cBase) NEW
 // Cargar la estructura de la base de datos
 aDbf := DBSTRUCT()
 // Crear la tabla deslizante
 oDbf := TBrowseDB( 1, 1, MAXROW() - 1,;
  MAXCOL() - 1 )
 //Añadir las columnas de cada campo
 AEVAL( aDbf, { | x | ;
  oDbf:AddColumn( TBColumnNew( x[ DBS_NAME ],;
  FIELDBLOCK( x[ DBS_NAME ] ) ) ) } )
 // Borrar la pantalla y dibujar el marco
 CLS
 DISPBOX( 0, 0, MAXROW(), MAXCOL(), 2 )
 // Personalizar tabla deslizante
 oDbf:HeadSep := "ÄÂÄ"
 oDbf:ColSep := " 3 "
 // Bucle de proceso
 WHILE .T.
  // Estabilizar el objeto
  oDbf:ForceStable()
  // Leer una tecla
  nTecla := INKEY( 0 )
  DO CASE
   CASE nTecla == K_ESC
    DBCLOSEALL()
    SET( _SET_COLOR, cColorAnt )
    CLS
    RETURN NIL
   CASE nTecla == K_UP
    oDbf:Up()
   CASE nTecla == K_DOWN
    oDbf:Down()
   CASE nTecla == K_LEFT
    oDbf:Left()
   CASE nTecla == K_RIGHT
    oDbf:Right()
   CASE nTecla == K_PGUP
    oDbf:PageUp()
   CASE nTecla == K_PGDN
    oDbf:PageDown()
   CASE nTecla == K_CTRL_PGUP
    oDbf:GoTop()
   CASE nTecla == K_CTRL_PGDN
    oDbf:GoBottom()
   CASE nTecla == K_HOME
    oDbf:Home()
   CASE nTecla == K_END
    oDbf:End()
   CASE nTecla == K_CTRL_HOME
    oDbf:PanHome()
   CASE nTecla == K_CTRL_END
    oDbf:PanEnd()
  ENDCASE
 ENDDO

RETURN NIL
```

CUADRO 1: Programa para visualizar cualquier base de datos mediante una tabla deslizante.

El cuadro 1 muestra el listado de un programa que visualiza el contenido de cualquier base de datos. El nombre y la ruta del fichero de base de datos debe ser pasado como argumento desde la línea de comandos del DOS.

```
DispBegin()
WHILE !oTabla:Stable .AND. NextKey() == 0
 oTabla:Stabilize()
ENDDO
DispEnd()
```

CONCLUSIÓN

Todo lo visto a lo largo de este artículo no es más que una pequeña parte de las posibilidades que ofrece la utilización de este tipo de objetos. Los objetos de esta clase aportan tal flexibilidad, que con ellos es posible realizar menús desplegables, visualizar matrices en forma de tabla deslizante, incluir ayudas on-line, e incluso realizar operaciones de proceso de textos. ■

REDES NEURONALES

Raúl Luna

Predicción de cotizaciones de bolsa, compresión de imágenes, filtrado de señales, etc. Estas son algunas aplicaciones de las redes neuronales en nuestros días. Sepa cómo funcionan.

Desde su aparición en 1943, las redes neuronales, que referenciaremos también como RN, han recorrido un tortuoso camino hasta nuestros días. Pasando de la fascinación inicial en sus comienzos, siguiendo por una etapa silenciosa desde 1965, hasta que en 1982-86 fueron rescatadas del olvido de la mano de Hopfield, momento en que comenzarían una ascensión que las ha situado como un tema fundamental de la investigación en nuestros días.

Pero ¿qué es una red neuronal? ¿en qué puede ayudarnos la programación con RN? ¿Mejorarían los resultados de un programa usando RN en lugar de las técnicas tradicionales?. Estas y otras preguntas se contestarán en el presente artículo.

Las RN surgieron en un intento de modelar matemáticamente un tejido neuronal. La cuestión era básica: si puedo construir un modelo matemático de un tejido neuronal que se comporta de un modo parecido a uno biológico, puedo saber qué funciones desempeña ese tejido en el ser vivo y cuales son los hechos realmente significativos del funcionamiento de una neurona. Además se puede experimentar por un medio mucho más barato -y menos cruel- que diseccionar animales.

Este fue el planteamiento que alimentó muchos trabajos sobre RN: inspirados en los tejidos nerviosos biológicos (al respecto, léanse los trabajos de Mc Culloch & Pitts y Hebb). Sin embargo, este planteamiento se fue olvidando por varias razones: el primero, que una neurona es un mecanismo mucho más complicado que las emulaciones que se construían y el segundo, el advenimiento a la investigación en el terreno de ingenieros, matemáticos e informáticos interesados sólo en el enfoque matemático y pragmático del asunto, dejando de lado la analogía biológica. Y así nos encontramos con que en nuestros días las RN son una colección de modelos matemáticos que se han comprobado resultan apropiados para resolver ciertos problemas, a saber: tratamiento de señal, visión artificial, predicción de series temporales, compresión de datos, extracción de características y un largo etcétera.

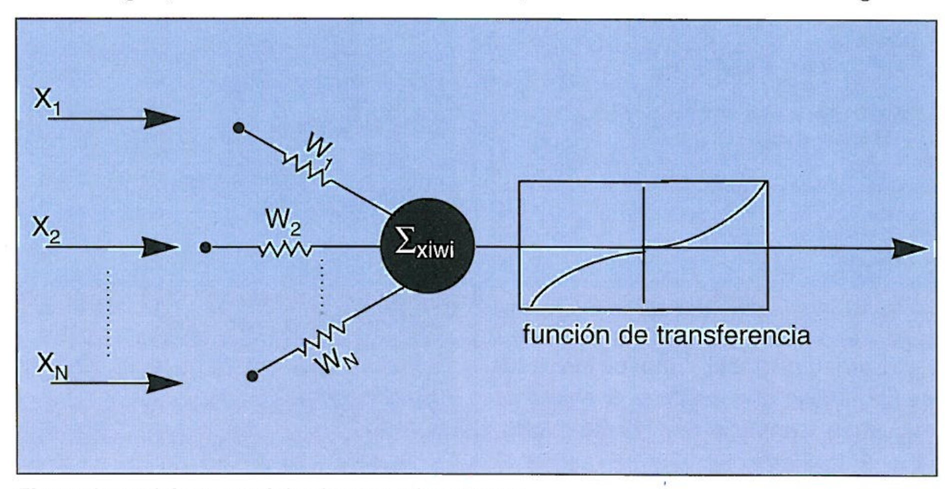


Figura 1: modelo general de elemento de proceso.

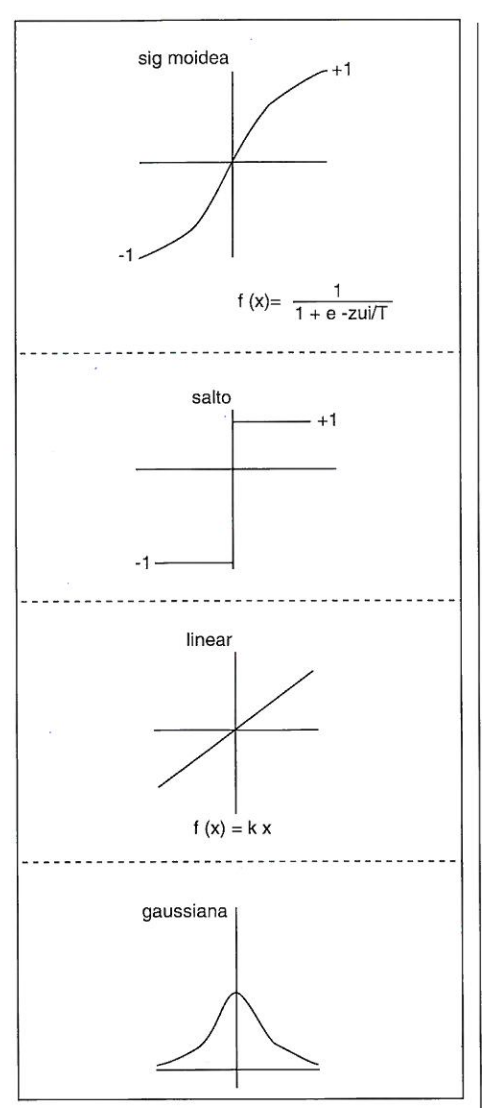


Figura 2: funciones de transferencia.

DESCRIPCIÓN GENERAL

Una red neuronal consta de una serie de Elementos de Proceso -abreviadamente EP- interconectados entre sí de tal modo que las salidas de unos constituyen las entradas de otros. Así, un EP recibe una serie de entradas provenientes de otros EP, realiza una serie de operaciones sencillas entre ellas y produce una salida, que irá a su vez a otros EP.

En la figura 1 puede verse un ejemplo: el EP -también llamado nodo o unidad- recibe una serie de entradas xi..xn en cada una de las cuales hay un peso wi i=1..n. El valor de entrada xi se multiplica por el peso. A continuación se suman todos los productos y se pasan a través de una función de salida f, que recibe el nombre de función de transferencia (ver figura 2), y que en el ejemplo corresponde a la sigmoidea.

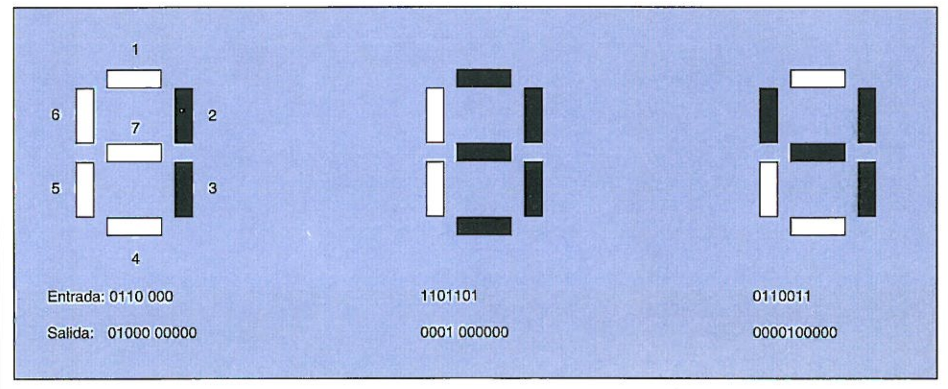


Figura 4: el problema del *display* de siete segmentos.

Las RN surgieron en un intento de modelar matemáticamente un tejido neuronal

Ocasionalmente se puede dotar a este EP de una memoria local , que suele consistir en el valor de salida que tuvo en su activación anterior: esto resultaría en una suerte de "memoria a corto plazo" que resultaría apropiada para resolver problemas de predicción de series temporales o de aprendizaje de secuencias de patrones. Ilustramos en la figura 3 cómo podría incluirse en nuestro EP. En lugar de incluir uno o dos registros en el interior del EP, se ha optado por incluir como entrada de la red en el instante t, las salidas en el instante t-1 y t-2, que nos vienen a través de los operadores de retraso z-1.

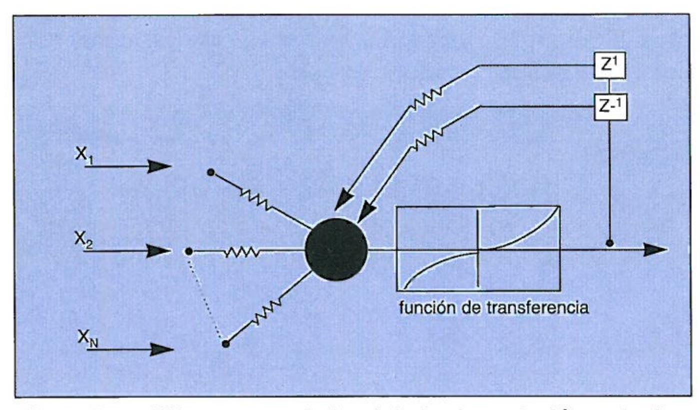


Figura 3: un EP con memoria local: la implementación se ha hecho conectando la salida a la entrada del EP mediante unas "cajas" de retraso.

APRENDIZAJE

Una diferencia significativa de las RN con respecto a la programación tradicional es su capacidad de aprender. El aprendizaje consistirá en que la red sea capaz de extraer una ley general en función de unos ejemplos, almacenar una serie de vectores o clasificarlos.

Distinguiremos dos tipos de aprendizaje: supervisado y no supervisado. El aprendizaje supervisado precisa que además de los ejemplos, indiquemos a la red que salidas esperamos para cada uno de ellos. Por ejemplo, si queremos entrenar una red para que sea capaz de leer un display de siete segmentos indicando qué segmentos están activados (figura 4) habría que suministrar a la red los segmentos que se han activado y la salida que se desea (por lo general que se active una neurona determinada).

Sin embargo, en el aprendizaje no supervisado se le suministran a la red solamente las entradas, y ella realiza tareas sencillas con ellos: agruparlos o almacenar esos patrones para evocarlos cuando se presente una parte del mismo.

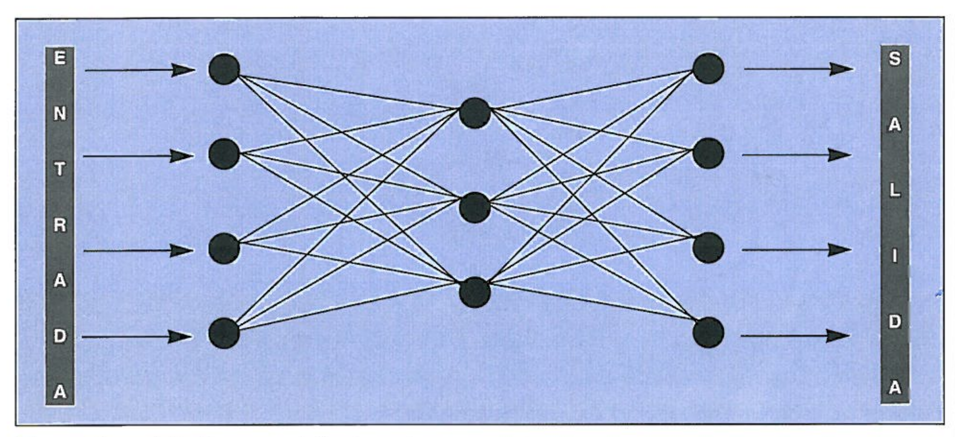


Figura 5: modelo de perceptrón de varias capas.

De este modo una red queda clasificada por los siguientes elementos:

- Topología: cómo se conectan los EP.
- Propiedades computacionales: descripción de los EP, reglas de activación, qué problemas puede resolver.

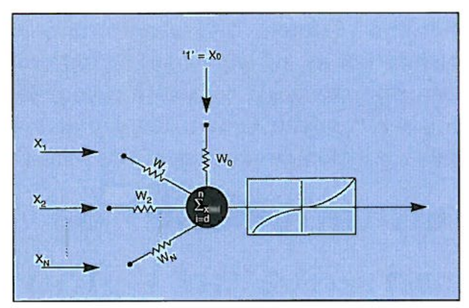


Figura 6: un EP del perceptrón.

- Regla de aprendizaje: de qué tipo es el aprendizaje y cuál es la regla de aprendizaje.

Pasaremos ahora a describir algunas de las redes más populares. Se incluye al final un listado de la implementación de uno de estos ejemplos.

RED PERCEPTRÓN

Este tipo de red se usa principalmente para problemas de clasificación e interpolación.

Topología

Los EP se agrupan por capas, conectándose todos-con-todos de una capa a la otra. No se permiten conexiones hacia atrás, ni entre elementos de la misma capa (ver figura 5). La primera capa es la de entrada y la última la de salida.

Descripción computacional

$$f(\sum_{i=0}^{n} x_i w_i)$$

Cada elemento de proceso calcula su salida conforme a la fórmula:

es decir, se suma el producto de cada entrada por el peso de esa entrada y el resultado se hace pasar a través de una función de transferencia, usualmente la sigmoidea. La entrada x0 está siempre a 1, de tal modo que el peso w0 siempre se suma y actúa como umbral.

Las RN son una colección de modelos matemáticos que, como se ha comprobado, resultan apropiados para resolver ciertos problemas

El proceso de cálculo comienza por la capa de entrada, luego se calculan las salidas de los EP de esa capa y se pasa a calcular los de la siguiente, y así sucesivamente hasta llegar a la última donde se produce la salida.

Para comprender cómo trabaja el perceptrón analicemos qué es lo que hace uno sólo de estos elementos con dos entradas (figura 7). Su salida sería f(x1w1+x2w2+w0). x1w1+x2w2+w0 es la ecuación de una recta en un plano con x1 en abcisas y x2 en ordenadas. La función de transferencia devolverá +1 si la entrada (x1, x2) se encuentra a un lado de la recta, y -1 si se encuentra en el contrario, clasificando de este modo las entradas.

Así, un modelo más general de perceptrón, con muchas más entradas y varias capas, trazará una serie de hiperplanos en un espacio vectorial de dimensión n que clasificará las entradas.

El problema típico que pueden resolver estas redes es el de clasificación. También son buenas para problemas de interpolación: a la red se la presentan como ejemplos pares (x, f(x)) de la función f(x) que se pretende aproximar.

Con unos ligeros cambios, estas redes pueden usarse en la predicción de series temporales (cotizaciones de bolsa, tiempo atmosférico,...). Por ejemplo, para las cotizaciones de bolsa puede recibir como entradas valores pasados de la cotización, y otros factores que se considere puedan tener interés -índice general, IPC, EPA,...- y como salida, la cotización que marcó ese día.

Es obvio que, para entrenarla precisamos una colección de datos históricos de bolsa.

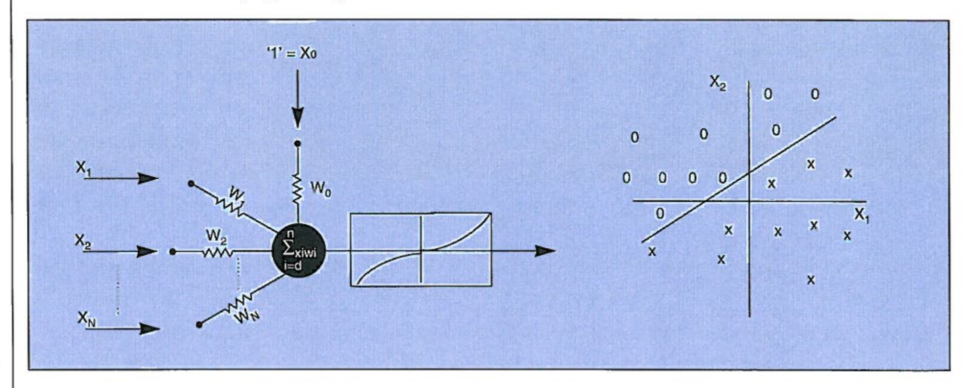


Figura 7: perceptrón de dos entradas e interpretación geométrica.




```
procedure BACK_PROP

inicializar los pesos a valores aleatorios pequeños
repeat
   tomar el siguiente patrón de entrenamiento (x, d);
   asignar como entrada de la capa 0 al vector x;
   FEED_FORWARD;
   CALCULAR_GRADIENTE;
   ACTUALIZAR_PESOS;
until que se alcance la condición de terminación;

end BACK_PROP;

procedure FEED_FORWARD

for capa=1 to L do
   for nodo=1 to N_capa do

         U_capa,nodo = f ( Σ_{i=0}^{N_capa-1} W_capa,nodo,i U_capa-1,i );

   endloop
endloop

end FEED_FORWARD;

procedure CALCULAR_GRADIENTE

for capa=L to 1 do
   for nodo=1 to N_capa do
      if capa=L then e_capa,nodo = u_capa,nodo - d_nodo;
         else

         e_capa,nodo = Σ_{m=1}^{N_capa+1} e_capa+1,m U_capa+1,m (1- u_capa+1,m) W_capa+1,m,nodo;

      endif
   endloop
   para todos los pesos en la capa "capa"

         g_capa,j,i = e_capa,j U_capa,j (1 - U_capa,j) U_capa-1,i;

   fpara;
endloop

end CALCULAR_GRADIENTE;

procedure ACTUALIZAR_PESOS

   para todo w_i,j,i hacer

         w_ij(k+1) = w_ij(k) - m g_ij

   fpara;

end ACTUALIZAR_PESOS;
```

Figura 8: algoritmo de retropropagación. Tomado de [Hush & Horne].

Aprendizaje, el algoritmo de retropropagación

En todas las RN el aprendizaje se realiza mediante un ajuste gradual de los pesos de cada uno de los EP. A la red se la presenta como entrada un vector dlÔ y se calcula su salida, que será un vector que llamaremos xOÔ. Ahora bien, la salida que se deseaba produjera viene dada por dOÔ: el error entre ambas se calcula mediante la función de residuos cuadrados, por medio de la fórmula:

$$\mathcal{E}=\frac{1}{2}\sum_{j}\left(d_{oj}^{\lambda}-x_{oj}^{\lambda}\right)^{2}$$

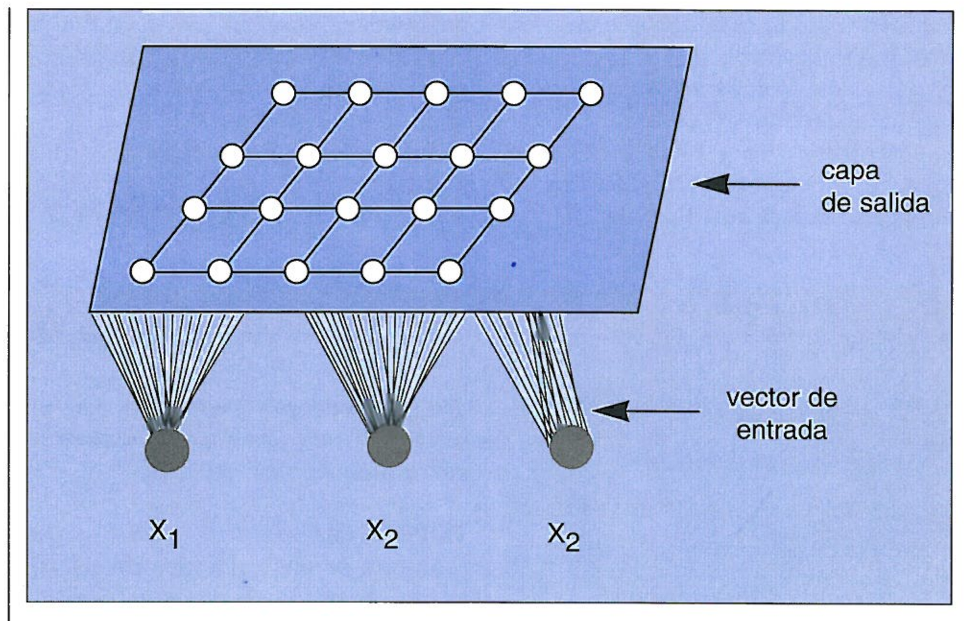


Figura 9: el conexionado de la red de Kohonen.

la medida de este error nos indicará cómo hemos de variar los pesos de las diferentes EP para ir acercando la salida de la red a la deseada. La función de actualización de pesos vendrá dada por:

$$w_{ij}^{l}(t+1)=w_{ij}^{l}(t)+\Delta w_{ij}^{l}$$

donde

$$\Delta w_{ij}^{l}=-\mu\frac{\partial\mathcal{E}}{\partial w_{ij}^{l}}$$

wijl es el i-ésimo peso de la unidad j-ésima de la capa l. El incremento de pesos consta de un valor real m - (0,1], que es la tasa de aprendizaje e indica la importancia que se le dará a la actualización de pesos. La política usual es disminuirlo progresivamente a medida que la red va aprendiendo

El aprendizaje se realiza mediante un ajuste gradual de los pesos

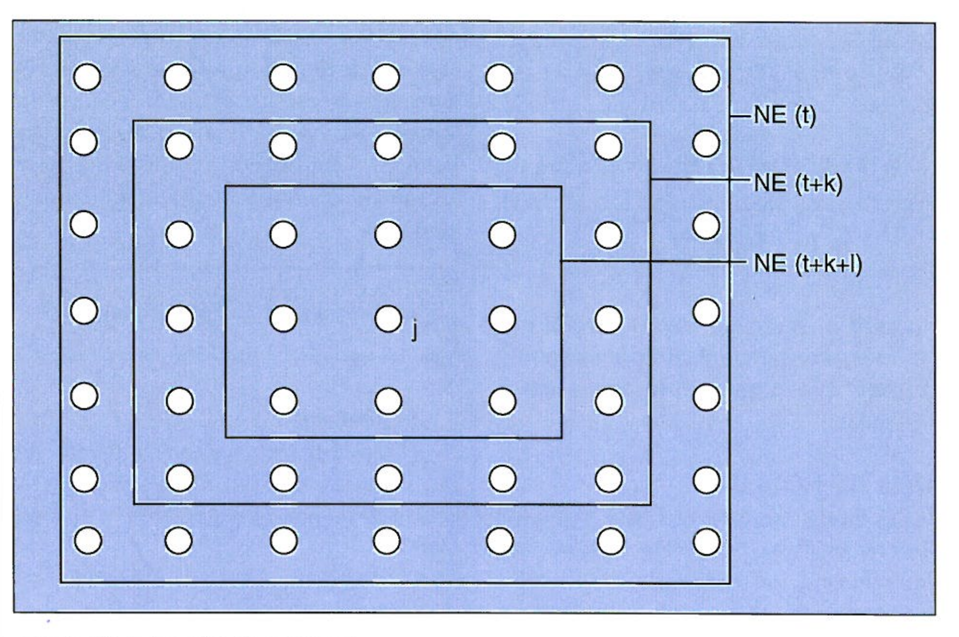


Figura 10: la función de vecindad.

los patrones. El resto de la expresión es la semiderivada de la función de error, en función de cada uno de los pesos a calcular. Operando convenientemente -véase bibliografía para una descripción detallada- se llega a una expresión de este tipo:

$$\Delta w_{ij}^{l\lambda}=\mu\delta_{i}^{l\lambda}x_{j}^{l-1}$$

```
procedure KOHONEN

  inicializar pesos con valores aleatorios;
  repeat
    presentar un nuevo vector de entrada (xi)
    mind = infinito; j*=0;
    para todos los nodos j

            di = Σ (xi (t) - wij (t) )²;

          if mind>dj then mind=dj; j*=j;
    fpara
    para todos los vecinos j de j*

     wij (t + 1) = wij (t) + m(t) (Xi (t) - wij 9t) );

    fpara
  until presentados todos los patrones


end KOHONEN;
```

Figura 11: algoritmo de aprendizaje para la red de Kohonen.

es decir, el producto de una función Ù por las entradas a esa capa, que pueden ser las salidas de la capa previa o las entradas suministradas a la red. La función Ù se calcula del siguiente modo:

$$\delta_{i}^{l\lambda}=(d_{0i}^{\lambda}-x_{i}^{l})\,g'(u_{i}^{l})$$

si la capa l es la última de la red y

$$\delta_{i}^{l\lambda}=g'(u_{i}^{l})\sum_{k}\delta_{k}^{(l+1)\lambda}w_{ki}^{(l+1)}$$

para el resto de capas. Incluimos en la figura 8 un algoritmo en pseudocódigo que implementa todas estas fórmulas.

RED DE KOHONEN

El diseño de esta red está inspirado en ciertas áreas del cerebro, en las que las neuronas se encuentran agrupadas en función de los estímulos que reciben. Por ejemplo, en el nervio auditivo las neuronas se agrupan en relación a la frecuencia que determina la mayor respuesta en cada neurona. Así se diseñó una red en la que los EP quedan agrupados conforme a una regla de vecindad, resultando muy adecuada para la resolución de problemas de optimización y cuantificación vectorial.

La capacidad de aprender del perceptrón, las diferencia de los métodos tradicionales

TOPOLOGÍA

Consta de una sola capa de salida, totalmente conectada con cada elemento de la entrada, como se puede ver en la figura 9.

Entre cada elemento de proceso se establecerá una relación de vecindad, que dotará a cada elemento de uno, dos o más vecinos. Por ejemplo, en la red de la figura 9 el número de vecinos es 4 menos los nodos que están en los bordes.

Descripción computacional

Todos los EP se actualizan a la vez. La salida de cada nodo se calcula de acuerdo a la regla:

$$x_{i}(t+1)=g(U_{i}(w_{i},x_{I}(t)))$$

donde Ui es una función que calcula una distancia entre el vector de entrada xl(t) y los pesos de ese nodo wi. Esta medida de distancia se pasa a través de una discontinuidad g que puede ser la función sigmoidea linear o de salto (figura 2). Una función de distancia muy usada es la norma euclídea, que viene dada por:

$$U_{i}=//x_{I}(t)-w_{i}(t)//_{E}=\sum_{j=1}^{n_{I}}(x_{Ij}(t)-w_{ij}(t))^{2}$$

Aprendizaje

El aprendizaje se puede calificar de no supervisado -no precisa indicar la salida deseada- y competitivo, como se verá.

En primer lugar se establecerá una función de vecindad, que señalará, para un elemento dado, un grupo de nodos vecinos. Esta función decrecerá con el tiempo (ver figura 10).

El algoritmo de aprendizaje se ilustra en la figura 11. Puede verse que tras presentar el vector de entrada, se calculan las distancias entre éste y los pesos de cada nodo, seleccionándose aquel que menor distancia tiene -que es, a la postre, el que mejor se aproxima a dicho nodo-. Para este nodo y sus vecinos se actualizan los pesos, aproximándolos aún más al vector presentado. Obsérvese que el vector que se elige para actualizarse es aquel que "gana" a los demás por tener el menor mínimo, de ahí la denominación de aprendizaje competitivo que indicamos al inicio de este apartado

EJEMPLO

Se incluye en el presente número de esta revista una implementación de una red de Kohonen. Se incluyen una serie de ficheros de datos los que se suministran valores de cotizaciones de bolsa "tipo": mediante estos datos se pretende que la red sea capaz de identificar unas figuras características de los mismos, y una vez entrenada, presentando un gráfico de cotización a la red, ésta sea capaz de identificarlo como perteneciente a un tipo dado. Este es uno de los fundamentos del llamado análisis técnico bursátil, que pretende detectar la tendencia de una cotización de bolsa por la forma que toma el gráfico de sus cotizaciones. ■

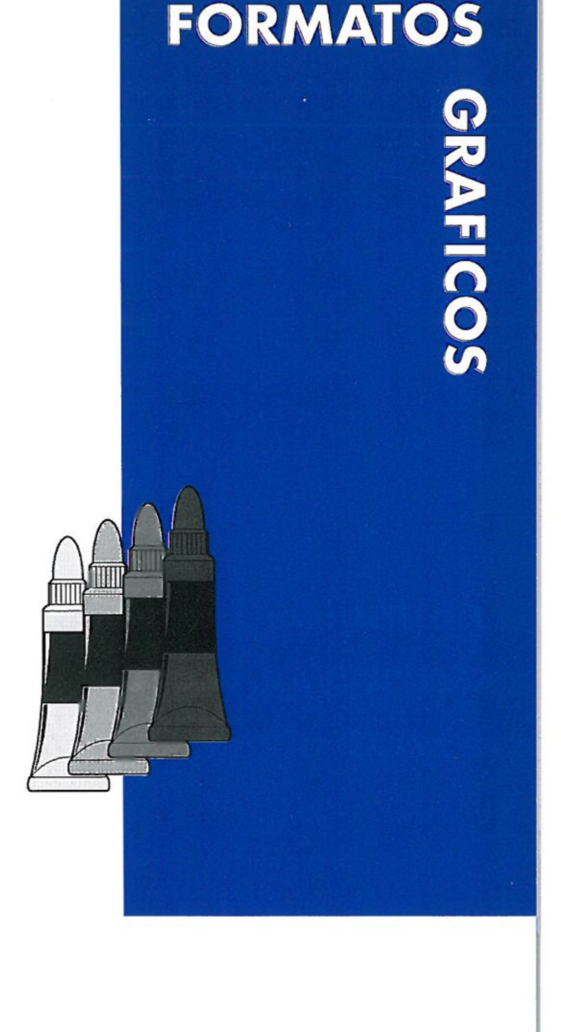
BIBLIOGRAFÍA:

R.P. Lippmann, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April 1987, pp. 4-22.

Hush, D. R. and B. G. Horne, Progress in Supervised Neural Networks: what's new since Lippmann, IEEE Signal Processing Magazine, January 1993, pp. 8-38.

Masson, E. and Y-J Wang, Introduction to computation and learning in artificial neural networks, European Journal of Operational Research 47 (1990) 1-28.

L.A.Merino, Curso de doctorado sobre Redes Neuronales, Univ. de Valladolid, Octubre 1994.



ESTÁNDAR ENTRE LOS ESTÁNDARES

Agustín Guillén

En esta ocasión se van a explicar las características más notables del formato gráfico TGA, su éxito se debe a su facilidad de uso, a la escasa cantidad de memoria necesaria para leer un fichero, y a ser el formato de color real (True Color) más extendido por el mundo. Prácticamente es el formato utilizado "por defecto", en la mayoría de los programas gráficos profesionales, bien sean de retoque de imágenes o de renderización.

INTRODUCCIÓN

En 1984, Truevision, la compañía creadora de las tarjetas gráficas profesionales más utilizadas: Targa, Targa + y la incombustible ATVista, creó por necesidades propias un nuevo formato gráfico capaz de almacenar las imágenes de 24, o de 32 bits creadas por su hardware. Pero en 1989, debido a las nuevas tecnologías y técnicas, tuvieron que ampliar la definición del formato, para poder incluir más información sobre la imagen almacenada, y para posibilitar el manejo de imágenes de gran tamaño. Algunas de las nuevas facetas que incluyeron son:

- La existencia de una copia reducida de la imagen, denominada "sello". Esta idea ha sido copiada muchos años después por Kodak en su formato de fotografías en CD.
- Fecha y hora de la creación de la imagen.
- Nombre y comentarios del autor.
- Descripción del trabajo y su duración.
- Nombre y versión del software que creó la imagen.
- Datos sobre los colores, canal alpha y aspect ratio de los pixels.
- Áreas definibles por el creador de la imagen.

Prácticamente es el formato utilizado "por defecto" en la mayoría de los programas gráficos profesionales

FORMATO

En la figura 1 se muestra la estructura de un fichero TGA. No tienen que estar necesariamente en este orden los distintos bloques de datos, pues algunos de ellos son referenciados desde un offset o desplazamiento a partir del inicio del fichero, y se deja la decisión sobre su ubicación final al desarrollador.

Un fichero TGA (la versión extendida, a partir de 1989) se compone de 5 áreas principales: cabecera TGA, datos de la imagen y del mapa de color, área del desarrollador o creador, área de extensión y el final de fichero TGA.

Lo primero que se debe comprobar, al leer un fichero TGA es, si se trata de la versión original o si tiene el nuevo formato. Esto se realiza leyendo los últimos 26 bytes del fichero (mediante la función SEEK, empezando por el final), si se trata de un fichero antiguo, encontraremos 26 bytes de datos de imagen, pero si es del tipo extendido, obtendremos el final de fichero TGA. Con estos 26 bytes en memoria, comprobaremos si entre los bytes 8 y 23 se

Desde 1984, cuando Truevision definió un nuevo formato gráfico para el uso de los primeros productos videográficos, el formato TGA se ha mantenido como el más utilizado en ambientes profesionales, comprime poco, pero su calidad y facilidad de uso lo han mantenido totalmente vigente.

encuentra la firma TGA en forma de la siguiente cadena:

TRUEVISION-XFILE

si esto aparece, se puede asegurar que se trata del nuevo formato y que puede contener el área del creador y/o el área de extensión.

A continuación detallaremos las distintas áreas con sus correspondientes campos:

CABECERA

Longitud del Identificador de la Imagen, campo 1 (1 byte)

Indica el número de bytes contenidos en el campo 6, el identificador de la imagen. Si aparece un 0, indica que no está incluido un identificador.

Tipo del mapa de color, campo 2 (1 byte)

0.- No hay un mapa de Color incluido en la imagen.

1.- Un Mapa de Color está presente en el fichero.

Las pantallas True Color no utilizan un mapa de color, por lo que este campo debe estar a 0, si en el campo 3 (tipo de imagen) encontramos una imagen True Color.

Tipo de imagen, campo 3 (1 byte)

Un fichero TGA puede almacenar imágenes True Color, Pseudo Color o Direct Color, y cada una de ellas puede ser a color o no. Los distintos valores posibles, son los siguientes:

Lo primero que se debe comprobar es si se trata de la versión estándar o de la extendida

0.- No existe ninguna imagen incluida.

1.- Imagen con mapa de color, no comprimida.

2.- Imagen True Color, no comprimida.

3.- Imagen en blanco y negro, no comprimida.

9.- Imagen con mapa de color, comprimida RLE (Run Lenght Encoded).

10.- Imagen True Color, comprimida RLE.

11.- Imagen en blanco y negro, comprimida RLE.

La codificación RLE se compone de dos tipos de paquetes diferentes: paquetes comprimidos o paquetes sin compresión. El primer byte de cada paquete es llamado contador de repetición y el segundo valor del píxel. Cuando es un paquete comprimido el valor del píxel es de un sólo byte, pero cuando se trata de un paquete sin compresión puede ser de hasta 127 bytes.

Paquete comprimido (RLE):

Nombre del campo	Desc. del campo	Tamaño del campo
Tipo de paquete	Debe ser 1	1 bit
Contador de Pixel	Núm.de pixels-1	7 bits
Valor del Pixel	El valor a repetir	Profundidad del Pixel (campo 5.5)

Paquete no comprimido:

Nombre del campo	Desc. del campo	Tamaño del campo
Tipo de paquete	Debe ser 0	1 bit
Contador de Pixel	Núm.de pixels-1	7 bits
Valor del Pixel	Los distintos valores	Profund. del Pixel * Contador de Pixel + 1

Especificación del Mapa de Color, campo 4 (5 bytes)

Este campo describe el Mapa de Color (si lo hubiera) utilizado por la imagen. Si la imagen no tuviera Mapa de Color, estos bytes deben estar a 0.

Índice de la Primera Entrada del Mapa de Color, campo 4.1 (2 bytes)

Indica a partir de que posición se debe acceder al Mapa de Color, esto permite utilizar solamente un número de colores reducido, dentro de un Mapa de Color extenso.

Longitud del Mapa de Color, campo 4.2 (2 bytes)

Número total de entradas en el Mapa de Color.

Tamaño de cada entrada en el Mapa de Color, campo 4.3 (1 byte)

Establece el número de bits por entrada. Suele contener 15, 16, 24 ó 32 bits.

Especificación de la Imagen, campo 5 (10 bytes)

Aquí se describen las características de la imagen, como su posición en pantalla o su tamaño.

-Origen X de la imagen, campo 5.1 (2 bytes)

Indica la posición horizontal inicial de la esquina inferior izquierda sobre la pantalla.

-OrigenY de la imagen, campo 5.2 (2 bytes)

Indica la posición vertical inicial de la esquina inferior izquierda sobre la pantalla.

-Anchura de la imagen, campo 5.3 (2 bytes)

Anchura de la imagen en pixels.

-Altura de la imagen, campo 5.4 (2 bytes)

Altura de la imagen en pixels.

-Profundidad de la imagen, campo 5.5 (1 byte)

Indica el número de bits por pixel de la imagen, incluyendo los bits del canal alpha.

-Descripción de la imagen, campo 5.6 (1 byte)

Bits 3-0: Especifican el número de bits de atributo para cada pixel o el número de bits para el canal alpha.

Bits 5 y 4: Indican el orden en el que los datos de cada pixel son transferidos desde el fichero a la pantalla. El bit 4 es para el orden de izquierda a dere-

Destino en la pantalla del primer pixel	Bit 5	Bit 4
abajo - izquierda	0	0
abajo - derecha	0	1
arriba - izquierda	1	0
arriba - derecha	1	1

cha y el bit 5 indica el orden de arriba a abajo:

Bits 7 y 6: Se deben poner a 0 para compatibilidad futura.

DATOS DE LA IMAGEN Y DEL MAPA DE COLOR

Identificador de la imagen, campo 6 (variable)

Se trata de un campo opcional que contiene información variada sobre la imagen. Su longitud máxima no puede sobrepasar los 255 bytes.

Datos del mapa de color, campo 7 (variable)

Estará presente sólo si el campo 2 (tipo del mapa de color) no es 0. Contiene la información referente al mapa de color. Con el campo 4.3 se indicará la anchura en bits de cada entrada y con el campo 4.2 el número total de ellas. Cada entrada está almacenada utilizando un número entero de bytes, por lo que aunque se especifiquen 15 bits como tamaño de la entrada (campo 4.3), ocupará 2 bytes completos, utilizándose los bits restantes como bits de atributos.

Datos de la Imagen, campo 8 (variable)

Este campo contiene los pixels resultantes de multiplicar la anchura por la altura. Cada pixel puede especificar los datos en alguno de los siguientes formatos: un índice para acceder al Mapa de Color (Pseudo Color), los valores rojo, verde, azul y atributos para cada pixel (True Color) o índices para Mapas de Color independientes (Direct Color).

ÁREA DEL CREADOR O DESARROLLADOR

Campos del desarrollador, campo 9 (variable)

Este campo se creó debido a las peticiones de los desarrolladores para almacenar sus propios datos e información. Su tamaño y formato puede variarse completamente de cara a las exigencias de los programadores y la mayoría de las aplicaciones que interpretan ficheros TGA, ignoran este área.

Como se puede incluir más de un campo en este apartado, se hace necesaria la presencia de un directorio del creador, que contiene un "mapa" de los campos incluidos en el área del creador. Este directorio se localiza gracias a un offset contenido en el área final del fichero TGA (bytes 4-7). Si el offset es 0, no existen ni directorio, ni área del creador.

En la figura 2, podemos observar el formato del directorio del creador. El primer valor (2 bytes) indica el número de tags o entradas de que se compone el directorio, y el resto del directorio contiene una serie de elementos compuestos por conjuntos de tag, offset y tamaño. Cada tag contiene un número de 2 bytes que indica el tipo de información que contiene el campo referenciado, el offset es un valor de 4 bytes (long) que señala el inicio del campo desde el principio del fichero y la parte del tamaño, señala la longitud en bytes del campo mediante 4 bytes (long).

Un fichero TGA puede almacenar imágenes True Color, Pseudo Color o Direct Color

ÁREA DE EXTENSIÓN

El área de extensión se añadió para poder incluir información extra sobre la imagen de manera estándar.

Su ubicación esta señalada mediante el offset del área de extensión, que se encuentra en el final del fichero TGA. Si este offset es 0, indica que no está incluida ningún área de extensión.

Tamaño de la Extensión, campo 10 (2 bytes)

Especifica el número de bytes del Área de Extensión. Actualmente su valor es fijo y es de 495, pero en futuras versiones (posteriores a la 2.0), este valor puede cambiar (incrementarse), por lo que conviene leerlo y tenerlo en cuenta de cara a procesar un fichero TGA.

Nombre del Autor, campo 11 (41 bytes)

Útil para guardar el nombre de la persona que creó la imagen, debe terminar en un 0 binario, por lo que restan un total de 40 bytes para el nombre.

Comentarios del autor, campo 12 (324 bytes)

Otro campo ASCII que se compone de 4 líneas de 80 caracteres, cada una de ellas seguida por un 0 binario. Aquí se suelen almacenar los diversos comentarios referentes a la imagen almacenada (tipo de prueba, como se capturó, etc.)

Fecha y Hora, campo 13 (12 bytes)

Se define en este campo la fecha y la hora en la que la imagen fue generada. El formato se compone de 6 shorts (número entero de 2 bytes) y tiene la siguiente forma:

Mes	1-12
Día	1-31
Año	4 dígitos (p.e. 1995)
Hora	0-23
Minuto	0-59
Segundo	0-59

Nombre del Trabajo / ID, campo 14 (41 bytes)

Campo ASCII que permite almacenar el nombre o código del trabajo que originó el fichero TGA.

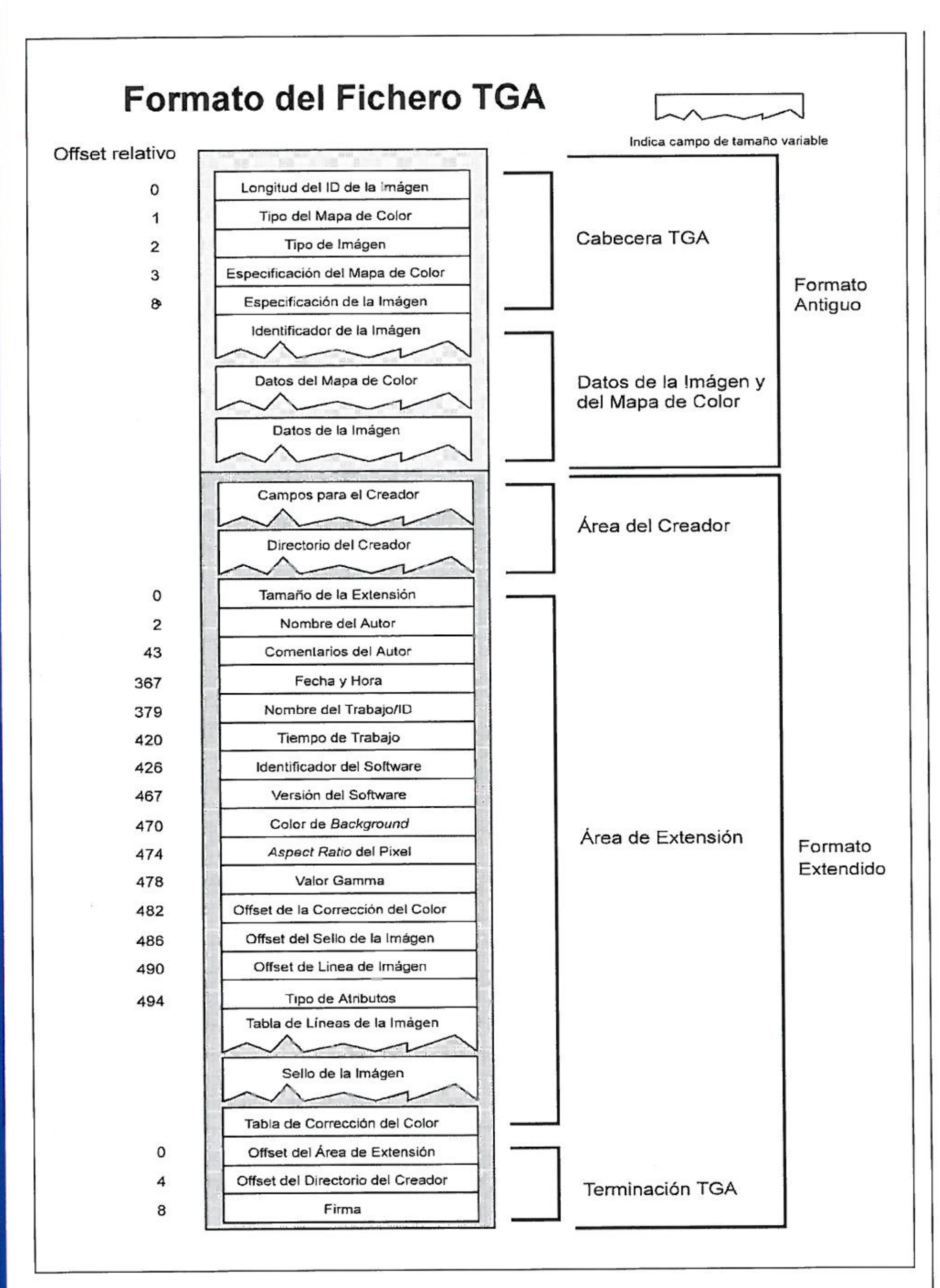
Tiempo del trabajo, campo 15 (6 bytes)

Aquí se guarda el tiempo que se tardó en generar la imagen (o el trabajo necesario para generarla) y tiene el siguiente formato:

Horas	0-65535
Minutos	0-59
Segundos	0-59

Identificador del software, campo 16 (41 bytes)

Sirve para informar sobre el software que generó la imagen, escribiendo en él su designación.



Versión del software, campo 17 (3 bytes)

Junto con el campo anterior, informa sobre la aplicación que creó el fichero TGA, tiene el siguiente formato:

bytes 0-1 Número de versión * 100 (p.e. 418 indica versión 4.18)

byte 2 Letra de la versión (p.e. "b" indica versión 4.18b)

Color de background, campo 18 (4 bytes)

Contiene un valor del tipo long que guarda el color de fondo o transparente, activo en el momento de guardar la imagen. Su formato es A:R:G:B (la letra A indica el canal alpha del color). Indica qué color de la imagen es "incoloro".

Aspect ratio del pixel, campo 19 (4 bytes)

Mediante dos valores, se especifica el aspect ratio de la imagen:

bytes 0-1 Numerador del ratio del pixel (anchura del pixel)

bytes 2-3 Denominador del ratio del pixel (altura del pixel)

Valor gamma, campo 20 (4 bytes)

De igual manera que en el campo anterior, el valor gamma se indica mediante dos valores:

bytes 0-1 Numerador gamma

bytes 2-3 Denominador gamma

El resultado debe estar comprendido entre 0.0 y 10.0, y una imagen sin valor gamma debe almacenar en este campo el valor 1.0 (ambos subcampos con el mismo valor) o tener en el denominador un 0.

Offset de la corrección del color, campo 21 (4 bytes)

Contiene un offset a la tabla de corrección del color, siempre contando desde el inicio del fichero. Poniendo un 0 en este campo, se indica que no existe tal corrección y que con el valor gamma es suficiente.

Offset del sello de la imagen, campo 22 (4 bytes)

Almacena el offset al sello de la imagen, si está a 0, indica que no existe tal característica.

Offset de línea de imagen, campo 23 (4 bytes)

Contiene el offset a la tabla de líneas de la imagen.

Tipo de atributos, campo 24 (1 byte)

Con este atributo se indica el tipo de canal alpha contenido en el fichero, sus valores pueden ser los siguientes:

0 Sin datos alpha (los bits 3-0 del campo 5.6 deben ser 0 también).

1 Datos indefinidos en el campo alpha, pueden ser ignorados.

2 Datos indefinidos en el campo alpha, pero deben ser guardados.

3 Canal alpha presente y utilizable.

4 Alpha premultiplicado.

5-127 Reservados.

128-255 Sin asignación.

Un valor de 4 (alpha premultiplicado) indica que el canal alpha ya ha sido precalculado previamente sobre los valores RGB.

Tabla de líneas de la imagen, campo 25 (variable)

En esta tabla se almacenan todos los offsets (4 bytes), para acceder directamente a las distintas líneas de la imagen. Es muy útil para poder leer cualquier parte de la imagen, aunque esté comprimida (tamaño variable) o para acceder fácilmente a pequeños trozos de una imagen muy grande. El

tamaño de la tabla será: altura de la imagen * 4.

Sello de la imagen, campo 26 (variable)

Se trata de una versión reducida de la imagen almacenada, es muy útil para visualizar rápidamente una aproximación de la imagen almacenada en el

bytes 0-3	Offset del área de extensión
bytes 4-7	Offset del directorio del creador
bytes 8-23	Firma
byte 24	Carácter ASCII "."
byte 25	Terminador de cadena (null, ascii 0)

fichero. Su formato será el mismo que el de la imagen principal, pero sin compresión. Los dos primeros bytes del campo indican la anchura y la altura del sello y se recomienda que no se sobrepase el tamaño de 64x64 pixels.

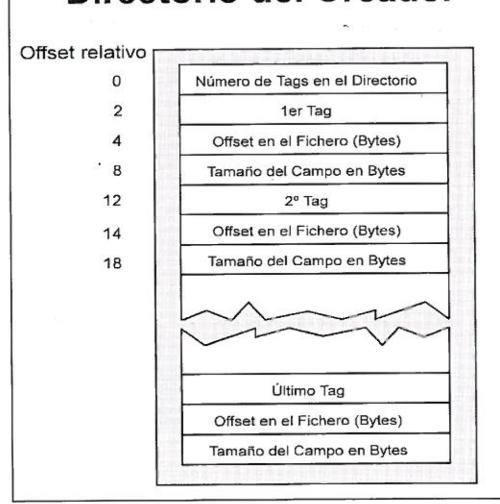


Tabla de corrección de color, campo 27 (2048 bytes)

Es un bloque de 256 x 4 shorts (enteros de 2 bytes), donde cada juego de 4 indica la corrección para esa entrada (A:R:G:B). Como cada color en el bloque es un short los valores oscilan entre 0 y 65535 (0:0:0:0 es negro total, 65535:65535:65535:65535 es blanco puro).

FINAL DEL FICHERO TGA

Como se ha dicho anteriormente, se compone de 26 bytes y tiene el siguiente formato:

Offset del área de extensión, campo 28 (4 bytes)

Indica donde se encuentra en inicio del área de extensión desde el inicio del fichero. Si aparece un 0, es que no existe un área de extensión.

Offset del directorio del creador, campo 29 (4 bytes)

Muestra donde se encuentra en inicio del directorio del creador desde el inicio del fichero. Si aparece un 0, es que no existe un directorio del creador.

Firma, campo 30 (16 bytes)

Se trata de una cadena ASCII, que indica si el fichero es de la nueva versión TGA (a partir de la versión 2.0) o se trata de un formato antiguo. Su formato es exactamente el siguiente: "TRUEVISION-XFILE" (sin comillas).

Carácter Reservado, campo 31 (1 byte)

Carácter ASCII necesario para identificar el fichero como un fichero válido TGA: "."

Terminador de cadena, campo 32 (1 byte)

Ascii 0, o null necesario para indicar el Final de fichero TGA, característica clásica en el tratamiento de cadenas en lenguaje C.

EJEMPLO PRÁCTICO

Para ilustrar toda esta información, se ha preparado un programa de ejemplo que lee un fichero TGA, accede a la información de su interior y la muestra en pantalla.

Nótese que los ficheros TGA, pueden tener las siguientes extensiones: WIN, VST, VDA o ICB, además de la clásica TGA.

GLOSARIO

Pseudo Color.- Los colores son conseguidos a través de un índice (paleta de colores) y están prefijados en la imagen o en la propia tarjeta gráfica.

El formato TGA fue creado por Truevision

True Color.- Cada valor de cada pixel contiene los valores RGB necesarios para obtener el color final.

Direct Color.- Es similar a estilo Pseudo Color pero los valores en los mapas de color pueden alterarse individualmente para el rojo, verde o azul. Cada entrada en la paleta almacena un valor RGB, con el que se accede a tablas independientes para cada componente. ■

BIBLIOGRAFÍA

Truevision TGA File Format Technical Specification
Versión 2.0
Truevision, Inc. 1991

ARQUITECTURA RISC

Francisco Monteagudo

Todos hemos oído hablar de ellos, sin embargo poca gente tiene una idea clara de lo que son. En este artículo explicaremos el significado exacto de este término, como son y en qué se distinguen de sus homólogos, los procesadores CISC.

En principio, RISC son las siglas de Reduced Instruction Set Computer, es decir, computador con juego de instrucciones reducido; y si preguntamos qué quiere decir esto exactamente, la respuesta que obtendremos es que, un RISC es un procesador con pocas instrucciones, que además deben ser sencillas para poder ejecutarse en un solo ciclo de reloj. Esta explicación resulta satisfactoria, hasta que empezamos a leer las especificaciones de diferentes micros, y descubrimos que es perfectamente normal que un RISC tenga cerca de 200 instrucciones, y que incluya funciones de cálculo en coma flotante; estas cifras resultan todavía más llamativas, si tenemos en cuenta que, por ejemplo, el Motorola 68000, que es un CISC, tiene menos de 100 instrucciones y además no tiene unidad de coma flotante. En este punto, la pregunta es inevitable, ¿Donde está la sencillez? es mas, si pueden tener tantas instrucciones como deseen y todo lo complejas que quieran, ¿En que se diferencian de los CISC? o mejor dicho ¿Se diferencian en algo?

EL PRIMER EXAMEN

Si efectuamos un examen mas detallado, en particular, si examinamos los lenguajes ensamblador de diferentes procesadores RISC, y los comparamos con los lenguajes ensamblador de diferentes CISC, empezaremos a ver diferencias. Lo primero que nos llamará la atención es que los RISC tienen un gran número de registros internos, lo normal son 32, y que las instrucciones aritméticas y lógicas sólo pueden operar sobre parámetros situados en registros, es decir, no existen modos de direccionamiento; por ejemplo, si queremos sumar dos valores que están almacenados en variables situadas en la memoria, primero tendremos que ejecutar unas instrucciones de lectura para copiarlos en registros y después ejecutar la instrucción de suma sobre dichos registros. Otra característica distintiva la encontraremos en la codificación binaria; en los RISC, todas las instrucciones tienen la misma longitud en bits. Llegados a este punto, es posible que algún lector astuto haya caído en la cuenta de que, puesto que el diseño del lenguaje máquina depende del diseño interno de la CPU, la diferencia CISC/RISC estaría en la forma de diseñar el chip, en la arquitectura del procesador; si alguien ha pensado esto, ha dado en el clavo.

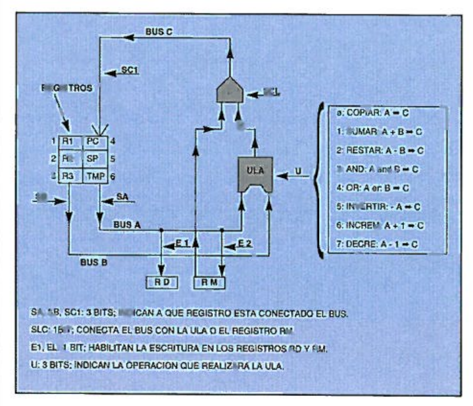
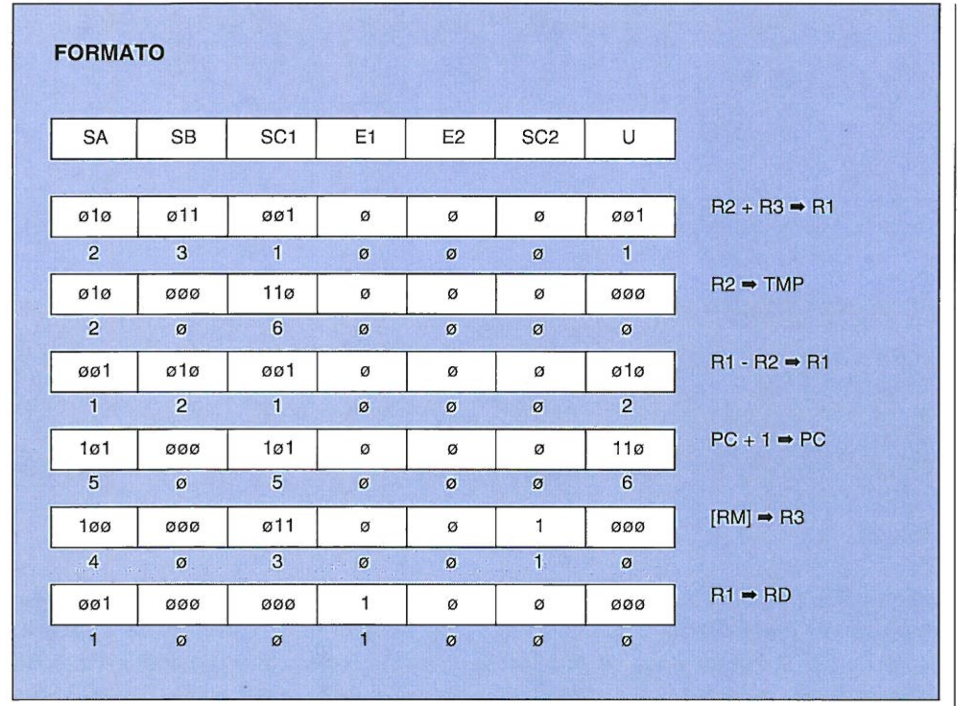


Figura 1.

La diferencia de la que hemos hablado está en la arquitectura interna, por eso es tan difícil explicarle a un profano que es un procesador RISC, porque este término no hace referencia a unas características técnicas concretas, sino a una filosofía de diseño, al método de trabajo seguido por el equipo de ingenieros encargados de diseñar el procesador; y por ese motivo, antes de intentar contestar a la pregunta que da título al artículo, vamos a explicar los principios básicos de arquitectura de procesadores.



FORMATO

SA	SB	SC1	E1	E2	SC2	U	
ø1ø	ø11	øø1	ø	ø	ø	øø1	R2 + R3 ➡ R1
2	3	1	ø	ø	ø	1	
ø1ø	øøø	11ø	ø	ø	ø	øøø	R2 ➡ TMP
2	ø	6	ø	ø	ø	ø	
øø1	ø1ø	øø1	ø	ø	ø	ø1ø	R1 - R2 ➡ R1
1	2	1	ø	ø	ø	2	
1ø1	øøø	1ø1	ø	ø	ø	11ø	PC + 1 ➡ PC
5	ø	5	ø	ø	ø	6	
1øø	øøø	ø11	ø	ø	1	øøø	[RM] ➡ R3
4	ø	3	ø	ø	1	ø	
øø1	øøø	øøø	1	ø	ø	øøø	R1 ➡ RD
1	ø	ø	1	ø	ø	ø	

Figura 2.

ANATOMÍA DE UNA CPU

Un procesador se divide en dos bloques principales, la unidad de proceso y la unidad de control.

LA UNIDAD DE PROCESO

Como se ve en la figura 1, la unidad de proceso está formada por la ULA (Unidad Lógica Aritmética), el juego de registros y los buses de comunicación.

LA ULA

La unidad aritmético lógica es el módulo de cálculo; está formada por un conjunto de circuitos independientes entre si, cada uno de los cuales realiza una operación especifica; hay un circuito para sumar y restar, otro para multiplicar, otro para invertir el signo de un número, etc. En el gráfico ya mencionado puede verse que dispone de dos entradas por donde se le entregan los valores a operar, una salida por donde entrega el resultado del cálculo pedido y unas líneas de control a través de las cuales se le indica a qué circuito estarán conectadas las entradas y la salida, qué operación se ejecutará; de esta forma, si deseamos realizar una suma, tendremos que colocar los valores que deseamos sumar en las entradas y activar la señal de control que dirige estos valores al circuito sumador, y ya podemos leer la salida, donde se encontrará el resultado de la operación.

LOS REGISTROS

Los registros son los encargados de almacenar las variables con las que el procesador está trabajando en ese momento y pueden dividirse en tres grupos: Los registros de usuario, que son los que están disponibles para el programador y pueden ser accedidos desde el lenguaje máquina; los temporales, que se usan para guardar los resultados intermedios que se generan durante la ejecución de instrucciones complejas; y por último, los registros de acceso a memoria; se trata de dos registros que se usan como canal intermedio para la lectura y escritura de datos en memoria, el primero, denominado RD (Registro de Dirección), almacena la dirección que saldrá por el bus de direcciones mientras que el segundo, denominado RM (Registro de Memoria), contiene el valor numérico leído o escrito a través del bus de datos. Los registros generales no suelen disponer de señales de control, aunque en algunos procesadores incorporan una señal de *reset* para ponerlos a cero. Normalmente, el registro PC (el contador de programa) incorpora una señal de autoincremento (cada vez que se activa, se suma 1 al valor del registro).

MICROPROGRAMAS

R1 ➡ RD [RM] ➡ TMP R2 + TMP ➡ R2	MICROGRAMA DE LA INSTRUCCION: ADD R2, [R1]
R1➡ RD [RM] ➡ TMP TMP + R2 ➡ TMP TMP ➡ [RM]	INSTRUCCION MAQUINA: ADD [R1], R2
R1 ➡ RD [RM] ➡ TMP R3 ➡ RD TMP ➡ [RM]	MOV [R3], [R1]
R2 ➡ RD [RM] ➡ TMP TMP + 1 ➡ TMP TMP ➡ [RM]	INC [R2]
R2 + R4 ➡ R2	ADD R2, R4
R3 + 1 ➡ R3	INC R3
SP ➡ RD R3 ➡ [RM] SP - 1 ➡ SP	PVSH R3
SP + 1 ➡ SP SP ➡ RD [RM] + ➡ TMP R2 ➡ RD TMP ➡ [RM]	POP [R2]

Figura 3.

LOS BUSES

Los buses son los encargados de mover los datos dentro del procesador; conectan elementos (a la manera de una tubería) transfiriendo información de uno a otro; el elemento al que está conectado cada extremo se decide mediante unas señales de control. En principio, cada bus necesita dos juegos de señales para decidir a qué elemento estará conectado cada extremo, pero en la práctica se prefiere que uno de ellos este enlazado permanentemente con la ULA; de esta forma solo necesitaremos un juego de señales para controlar el extremo libre. En el gráfico de ejemplo, el único bus que tiene señales para controlar ambos extremos es el C.

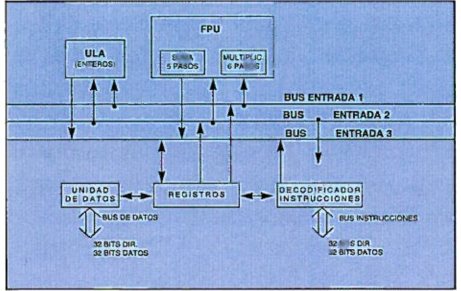
LA UNIDAD DE CONTROL

Hemos visto que la unidad de proceso se gobierna mediante unas señales; así, si en la unidad de proceso de la figura 1 queremos sumar el contenido de dos registros R1 y R2, y guardar el resultado en el registro R3, tendremos que fijar en las señales SA y SB los índices de los registro R1 y R2, en la señal SC1 el índice al registro R3, en la señal SC2 el índice a la ULA y en la señal U el código de la operación suma; es decir, la realización de una operación completa, como la descrita, requiere activar de forma simultánea un conjunto de señales de control; a esta acción se la denomina microinstruc-

ción. Dicho con otras palabras, la unidad de proceso se gobierna mediante microinstrucciones, cada una de las cuales define una operación realizable en un solo paso, que operan de una forma similar a las instrucciones de cualquier otro lenguaje; pero, ¿quién genera estas microinstrucciones? ¿quién le dice a la unidad de proceso lo que tiene que hacer? Esta es la función de la UNIDAD DE CONTROL; su misión, a grandes rasgos, es leer las instrucciones de lenguaje máquina de la memoria y convertirlas en microinstrucciones ejecutables por la unidad de proceso.

CODIFICACIÓN DE LAS MICROINSTRUCCIONES

Pero vayamos por partes; en primer lugar hay que ver como se codifican las microinstrucciones; estas no dejan de ser un conjunto de terminales eléctricos pueden estar activados o desactivados según lo que se desee hacer; esto significa que podemos considerar todo el conjunto de señales como un conjunto de bits, como un registro cuyos campos son las señales concretas; esto se ve claramente en la figura 2, donde se muestra el formato de lo que se podría denominar el registro de microinstrucciones de la unidad de proceso de



Figuras.

ejemplo y la codificación numérica y mnemotécnica de algunas microinstrucciones. Esta representación permite manejarlas como un lenguaje de programación, lo que hace posible la construcción de microprogramas que implementarán las instrucciones de lenguaje máquina mas complejas como secuencias de microinstrucciones; esto se puede ver en la figura 3.

Ahora que sabemos como se trabaja con las microinstrucciones, queda por ver que ocurre con las instrucciones del lenguaje ensamblador (este apartado se refiere a los procesadores CISC). El primer paso es leer un bloque de bits de la memoria, el segundo paso es analizar estos bits para detectar una instrucción; este último paso es necesario porque, como se mencionó al principio del artículo, las instrucciones máquina de estos procesadores tienen una longitud variable; el tercer paso es el análisis de la instrucción para reconocer su formato (el formato se refiere al significado de los bits dentro de la instrucción) y, de esta forma, saber como deberá decodificarse; el cuarto paso es la decodificación de la instrucción; es en este último paso donde se realiza la conversión en microinstrucciones; concretamente, se debe distinguir si la instrucción puede convertirse directamente en una microinstrucción o bien si tiene un microprograma asociado; es decir, hay que distinguir si la instrucción es SIMPLE o COMPLEJA; si es simple, solo queda construir la microinstrucción pertinente y enviarla a la unidad de proceso, y si es compleja, hay que localizar el microprograma que le corresponde y ejecutarlo.

	6 BITS	5 BITS	5 BITS	11 BITS	5 BITS
1	TRIADIC	D	S 1	OP CODE	SL
2	BIT	D	S 1	OP | ANCHO	OFFSET
3	INMED	D	S 1	INMEDIATO	
4	CBR	M/B	S 1	OFFSET	
5	BR/BSR	DESPLAZAMIENTO			

1: Instrucciones matemáticas con tres registros.
2: Operaciones con campos de bits.
3: Operaciones con valores inmediatos.
4: Operaciones de saltos condicional.
5: Operaciones de salto incondicional/llamada a subrutina.

Figura 4.

LOS PROBLEMAS DE LOS PROCESADORES CISC

Como hemos visto, dentro de un procesador quien realmente realiza el trabajo es la unidad de proceso, mientras que la unidad de control realiza tareas administrativas, tareas que, como se ha visto, añaden pasos al proceso de decodificación y ejecución. El resultado de todo esto es una disminución en la velocidad de ejecución (más ciclos de reloj por instrucción) y un aumento en el tamaño de la unidad de control. Además, queda otro problema añadido; como puede verse en la figura 3, no basta un microprograma por cada instrucción compleja, sino que cada variante de dicha instrucción requiere el suyo; por ejemplo, si un procesador tiene 100 instrucciones matemáticas y dispone de 5 modos de direccionamiento (cada instrucción tiene 5 variantes), tendremos que la unidad de control debe almacenar unos 500 microprogramas.

Después de estas explicaciones, podemos suponer que la unidad de control debe ocupar un espacio muy importante dentro de la CPU, y de hecho lo ocupa; así, en los CISC de 32 bits típicos (los 80386, 68030, etc.) unidad de control representa, según el procesador, entre el 70% y el 90% de la circuitería; así, de los 350.000 transistores que tiene un 80386 (aproximadamente), la unidad de control ocupará cerca de 300.000. Se podría decir que un CISC es como una fábrica, en la que hay más directivos que trabajadores, es evidente que su funcionamiento tiene que ser poco eficaz.

Y, POR FIN, LA AUTÉNTICA DEFINICIÓN DE RISC

Después de estas explicaciones ya podemos ver cual es la auténtica definición de RISC: Un RISC es aquel procesador en el que la unidad de control se ha simplificado al máximo. Para ser mas precisos, se consideran RISC aquellos procesadores cuya unidad de



control representa menos del 50% de la circuitería, y en la práctica ésta suele ocupar en torno al 20%. De esta forma, si un procesador RISC tiene 350.000 transistores (como el 80386) su unidad de control consumirá menos de 50.000, quedando unos 300.000 para la unidad de proceso. En principio, la circuitería que se ahorra tiene como destino aumentar al máximo el número de registros, aunque nada nos impide incorporar módulos funcionales extra, como por ejemplo un calculador en coma flotante.

REALIZACIÓN PRACTICA

De lo dicho anteriormente se deduce, que el objetivo de la tecnología RISC es reducir la "burocracia" dentro del procesador, para poder dedicar la circuitería a tareas "productivas". Lograr este objetivo requiere diseñar el lenguaje máquina del procesador, de manera que la decodificación de las instrucciones sea lo mas sencilla posible. Para lograrlo se utiliza una combinación de técnicas; la primera, y más importante, es la eliminación de las instrucciones complejas, que como se ha visto anteriormente, son aquellas que llevan asociado un microprograma; esto significa que debe existir una relación directa entre las instrucciones del lenguaje máquina y las microinstrucciones de la unidad de proceso, cada instrucción máquina debe corresponderse con una microinstrucción; la segunda técnica es hacer que todas las instrucciones tengan la misma longitud en bits, longitud que, además, será la del bus de datos; y la tercera técnica consiste, en reducir al máximo el número de formatos de instrucción; además, las instrucciones estarán divididas en dos partes; la primera parte será un campo de bits que indica el formato y la segunda parte será la instrucción propiamente dicha.

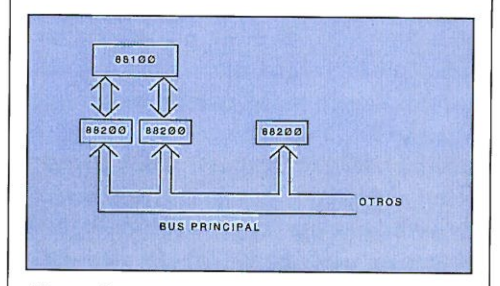


Figura 6.

EJECUCIÓN DE UNA INSTRUCCIÓN RISC

Con estas técnicas, la decodificación de una instrucción queda reducida a, en primer lugar, leer la instrucción de la memoria (como que la longitud de las instrucciones es la del bus de datos, una lectura de memoria es una lectura de una instrucción), en segundo lugar se comprueba el valor del campo formato, y se envía la instrucción al decodificador que corresponda a ese formato, y en tercer lugar, el decodificador mapea los bits de la instrucción sobre las señales de control de la unidad de proceso, ejecutándola. Como vemos, el proceso de decodificación y ejecución se ha simplificado enormemente, lo que supone, además del mencionado ahorro de transistores, un importante ahorro de tiempo. En la práctica, las instrucciones se ejecutan en un sólo ciclo de reloj.

Después de estas explicaciones, quedan aclaradas las diferencias en el lenguaje máquina explicadas al principio de este artículo, aunque es necesario explicar que la exigencia de que los parámetros de las instrucciones matemáticas estén en registros, obliga a disponer de ellos en un número importante, para que todas las variables que necesite la rutina en ejecución se puedan mantener en el interior del microprocesador, y de esta forma reducir al máximo las instrucciones de lectura/escritura en memoria; en principio, estas últimas solo deberían ser necesarias en la entrada a la rutina (para cargar las variables en los registros) y a la salida (para recopiarlas a memoria).

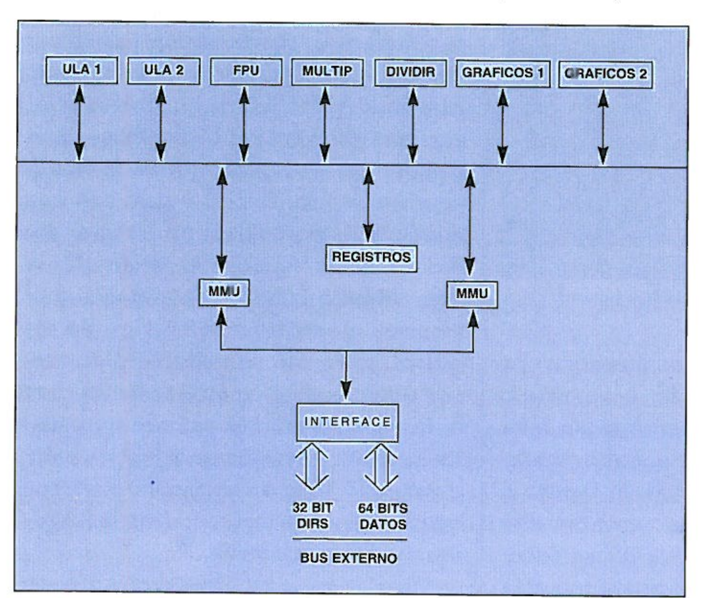


Figura 7.

CONSECUENCIAS DE ESTE DISEÑO

Una vez explicadas las diferencias entre los lenguajes ensamblador de los procesadores CISC y RISC, es necesario explicar qué repercusiones tienen éstas a la hora de programar.

La forma mas evidente de ver estas diferencias es mediante la comparación de rutinas en ensamblador. El método elegido en este artículo ha sido la confección de una pequeña subrutina en lenguaje C, CRIBA.C, que graba en el *array* salida los 100 primeros números primos, que calcula utilizando el algoritmo de la criba de Eratóstenes. Este fuente ha sido compilado en un PC 80486, CRIBA.386, y en un Sparc 20 CRIBA.SPR; en ambos casos el compilador utilizado fue el GNU C, compilador que se caracteriza por estar disponible para un gran número de máquinas diferentes (tanto CISC como RISC) y permite generar fuentes en ensamblador; en ambos casos la compilación se hizo con las optimizaciones activadas.

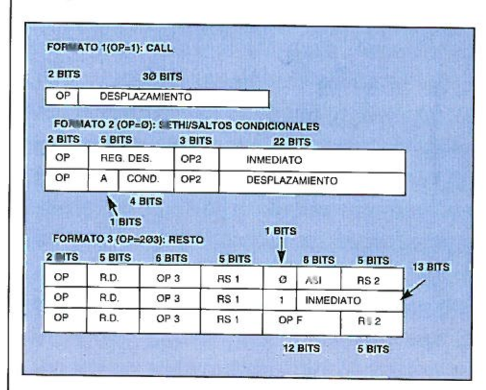


Figura 8.

Antes de comentar el listado, es necesario aclarar algunos conceptos. En

ambos listados los registros van precedidos por el signo "%"; así, se puede ver que en el código del 486 los registros utilizados son EAX, EBX, ECX, EDX, ESI y EDI, mientras que en el Sparc son L0, L1, L2, L3, L4, O0, O1 y O2. En el listado del 486 también se observan, a la entrada y la salida de la rutina, unas instrucciones push y pop para preservar los valores de los registros EBX, ESI y EDI durante la ejecución. En el Sparc

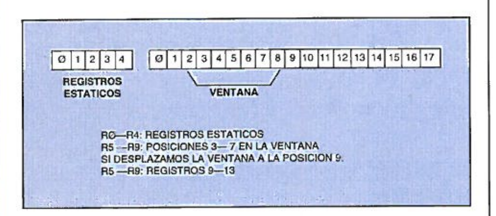


Figura 9.

esto se realiza mediante las instrucciones save y restore; su funcionamiento exacto se explica mas adelante, en el apartado dedicado a este procesador.

COMPARACIÓN DE PROGRAMAS EN ENSAMBLADOR

Al comparar ambos listados en ensamblador, saltan a la vista todas las características detalladas anteriormente; así, en el Sparc todas las variables están en registros, mientras que en el 486 la variable o se ha tenido que situar en memoria debido a la falta de registros; también puede observarse que en el 486 hay instrucciones que usan parámetros situados en memoria, cosa que el Sparc no hace. Pero lo más llamativo es la forma como se trabaja con los valores inmediatos (los valores contenidos en la misma instrucción máquina); en el Sparc las dos primeras asignaciones a salida se hacen en dos pasos; una primera instrucción carga el valor en un registro, y en la siguiente instrucción ese registro se copia en la posición destino; además, el proceso de carga de la posición del array salida también se hace en dos instrucciones, primero se ejecuta una instrucción sethi que carga los 16 bits altos [%hi(salida)] y después una instrucción or que añade los 16 bits bajos [%lo(salida)]. Esto último también es consecuencia de una característica de los procesadores RISC, la obligación de que las instrucciones tengan longitud fija. En el Sparc, como en la mayoría de las CPU RISC, las instrucciones tienen 32 bits de longitud; esto significa, que el valor inmediato más grande que se podrá almacenar estará, según el formato de la instrucción, en torno a los 20 bits (22 bits en el Sparc); si tenemos en cuenta que sus registros internos también son de 32 bits, nos encontramos con que, si queremos cargar una dirección de memoria en un registro (un valor de 32 bits), lo tendremos que hacer en dos pasos. De hecho, esta es la función de la instrucción sethi, cargar un valor en los 16 bits altos del registro, mientras que la instrucción mov escribe en los 22 bits bajos. Además, tampoco será posible introducir instrucciones que escriban valores inmediatos en memoria, porque no disponemos de suficientes bits para todos los datos necesarios.

Después del examen, queda claro que el programa para 486 tiene menos instrucciones y ocupará bastante menos memoria, pero si convertimos este código en microinstrucciones, veremos que el Sparc será algo mas rápido; la causa está en todas las instrucciones del 486 que operan sobre memoria, es decir, que se desglosan en varias microinstrucciones.

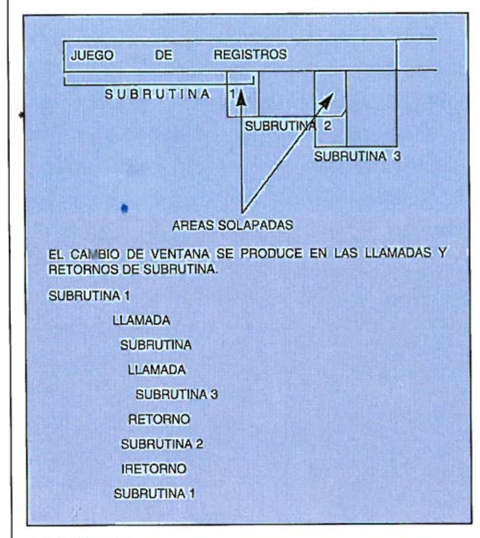
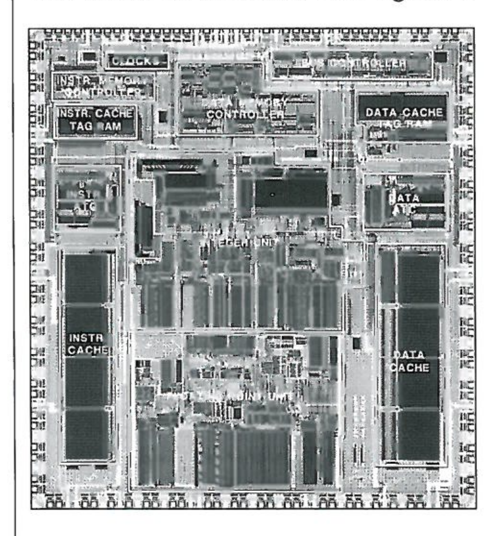


Figura 10.

Queda otro detalle que comentar; ya se ha dicho que, en el 486, la variable o se ha almacenado en memoria por falta de registros, pero, ¿por qué esta variable y no otra? si hacemos un repaso a este programa, seguramente se nos ocurrirán multitud de variaciones sobre el código; unas harán el programa más rápido, otras lo harán más corto, lo evidente es que el compilador podría haber generado el código de otras muchas formas diferentes; curiosamente, en el código para Sparc esta situación no se da; por más que examinemos el programa, es difícil que se nos ocurra otra forma de escribirlo. Esta rigidez a



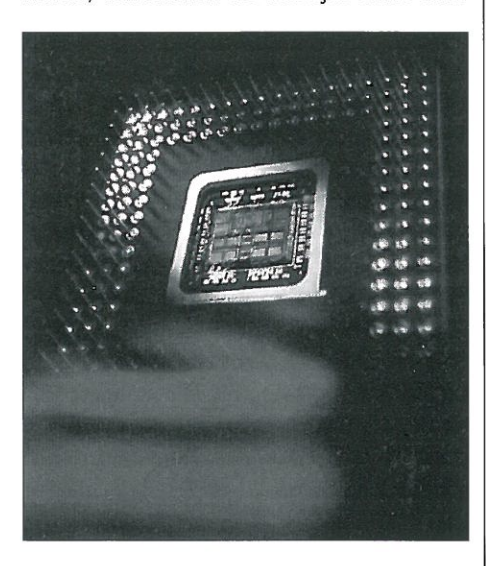
la hora de programar es también una característica de los procesadores RISC, y tiene una consecuencia muy importante: escribir un compilador para un procesador RISC es mucho más fácil, que hacerlo para un procesador CISC.

INCONVENIENTES DE LA TECNOLOGÍA RISC

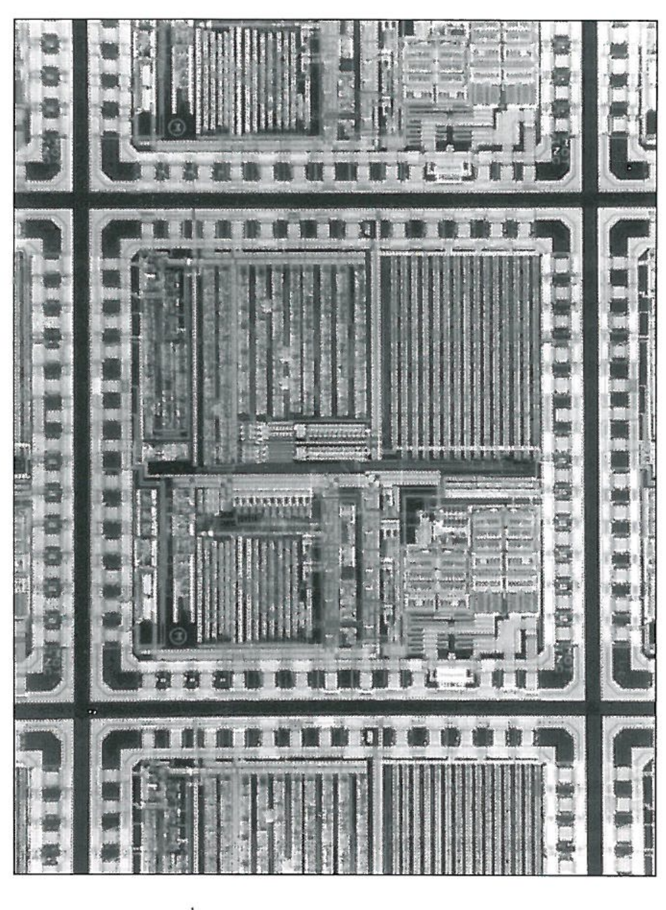
Después de estas explicaciones, resulta evidente que la tecnología RISC tiene muchas ventajas; sin embargo, todos sabemos que en este mundo nada es perfecto, por lo que inmediatamente supondremos que también debe tener inconvenientes, y ciertamente los tiene. El primer inconveniente viene del hecho, de que todas las instrucciones tengan la misma longitud, que en la práctica suele ser de 32 bits; a esto hay que añadir la imposibilidad de trabajar sobre variables situadas en memoria, lo que obliga a añadir instrucciones adicionales para cargar valores en los registros; todo esto se traduce en un mayor tamaño del código resultante, con lo que los programas ocupan más memoria. Este problema no tiene solución, aunque no se le da importancia porque actualmente los ordenadores incorporan memoria suficiente.

Otro importante inconveniente es, paradójicamente, consecuencia de su

mayor ventaja; el poder ejecutar una instrucción en cada ciclo de reloj tiene como consecuencia, que el procesador tiene que acceder al bus constantemente, con lo que este queda colapsado, provocando importantes retardos, en cualquier otro dispositivo que necesite utilizarlo. Por fortuna, este problema tiene una solución sencilla, añadir una memoria caché al microprocesador, lo que permite reducir el número de accesos a unos niveles razonables. Esta estrategia se ha llevado aún mas lejos añadiendo dos cachés, uno para las instrucciones y otro para los datos; esta técnica supone dotar a la CPU de dos interfaces independientes, entre los buses internos y el bus externo (los dos cachés), lo que en la práctica equivale a tener dos buses externos; esto permite que las instrucciones de lectura y escritura puedan ejecutarse en un solo ciclo de reloj, ya que el acceso a memoria no tiene que esperar a que el decodificador de instrucciones libere el bus. De todas maneras, esta forma de trabajar tiene una



limitación; puesto que los cachés son independientes entre si, en el caso de que señalen a las mismas posiciones de memoria podrían darse inconsistencias; por ejemplo, consideremos que hay una posición de memoria que contiene el valor 5; al acceder a esta posición desde los dos interfaces, se copia en los dos cachés; si posteriormente el procesador realiza una escritura y la cambia a 8, esta escritura se hará a través del caché de datos, y de este irá a la memoria, pero no al caché de instrucciones, con lo que seguirá conservando el valor anterior, el 5. La consecuencia de todo esto es que, en los procesadores RISC no están permitidos los programas automodificables.



Queda un tercer problema que también es consecuencia de una ventaja. Ya ha quedado clara la importancia y las ventajas de tener un número elevado de registros, sin embargo, en un sistema multitarea esto se puede convertir en un problema. En el momento de cambiar de un programa a otro, la CPU tiene que guardar en memoria, el contenido de todos los registros del programa del que sale, y leer de memoria el contenido de todos los registros del programa en el que entra. Si el procesador tiene 8 registros (como el 80836), esto no supone ningún problema, pero si tiene 32, 64 o 128 registros (valores típicos en el mundo RISC), entonces el proceso de cambio se ralentiza excesivamente, lo que puede reducir de forma visible el rendimiento global del sistema. Por desgracia, este problema no tiene una solución sencilla; se han experimentado diversos métodos para resolverlo, pero los resultados no terminan de ser satisfactorios.

LA RISCSIFICACIÓN DE LOS CISC

Pese a sus inconvenientes, es evidente que los microprocesadores RISC son muy superiores a los CISC; está claro que acabarán convirtiéndose en el estándar. Sin embargo, existe un gigantesco parque de máquinas CISC ya instaladas (fundamentalmente los PC basados en procesadores x86), y lo que es más importante, el grueso del software que se encuentra en el mercado también está diseñado para estas máquinas, por lo que los usuarios se niegan a cambiar; este cambio supondría tener que empezar de cero. Esto significa que los sistemas existentes tienen garantizada su supervivencia durante los próximos años, y por ello los fabricantes siguen lanzando al mercado procesadores compatibles con los CISC clásicos. Sin embargo, el que la vieja tecnología tenga garantizada su supervivencia no significa que pueda ser inferior, y por ello los diseñadores se esfuerzan por mejorar sus prestaciones al máximo para conseguir igualar a los RISC. La estrategia adoptada para lograr este objetivo es bien simple, imitar al enemigo; es evidente que la necesaria compatibilidad del código ejecutable impide la reconversión de los viejos procesadores a la nueva tecnología, pero nada impide adoptar algunas estrategias propias de los procesadores RISC, como el uso de uno o dos cachés. En la práctica se ha llegado mucho mas lejos; así, los últimos modelos de microprocesadores CISC tienen un diseño en dos niveles; internamente son procesadores RISC, a los que se les ha acoplado un circuito traductor, que convierte las

instrucciones leídas desde la memoria en instrucciones del mencionado micro. El resultado son unos procesadores híbridos. Esta es la estrategia utilizada en últimos miembros de la familia x86 y compatibles.

UN EJEMPLO PRÁCTICO, LA FAMILIA 88000 DE MOTOROLA

Motorola entró en el mundo RISC con esta familia de procesadores. Esta formada por los microprocesadores 88100 y 88110, y por el 88200, chip que incorpora una memoria de 16 Kb de caché. Estos procesadores tienen un juego de instrucciones compatible (el del 88110 es una ampliación del 88100); como se puede ver en la figura 4, dispone de 5 formatos de instrucción, de una longitud de 32 bits. Otra cosa que tienen en común, es el juego de registros, ambos tienen un total de 64 registros, 32 de ellos se utilizan para almacenar enteros, y tienen una longitud de 32 bits, y los otros 32 tienen una longitud de 80 bits y almacenan valores en coma flotante.

El 88100 es el hermano pequeño; como se puede ver en la figura 5, incluye una ULA para cálculo de enteros y una FPU (unidad de cálculo en coma flotante) capaz de sumar y multiplicar, aunque su característica más llamativa es que dispone de dos buses externos, uno para la lectura/escritura de datos en memoria, y otro para la lectura de instrucciones. En la figura 6 se puede ver la arquitectura típica del 88100; el procesador tiene conectados dos 88200 que actúan como interfaz y caché de la memoria principal; este montaje es, además, un ejemplo práctico del ya mencionado modelo de doble caché. El 88100 cuenta con otra característica destacable y es su capacidad superescalar; la FPU funciona como un coprocesador independiente, de forma que mientras está ejecutando una instrucción, el procesador puede seguir ejecutando código; además, como la FPU tiene una arquitectura en pipeline, puede iniciar la ejecución de una nueva instrucción, aunque aun no haya terminado la anterior. Resumiendo, el 88100 es un procesador que combina una gran sencillez conceptual, con unas excelentes prestaciones.

¿Y el 88110? Como se ve en la figura 7, tiene un sólo bus externo, aunque internamente incorpora dos cachés independientes; también podemos ver que su bus de datos es de 64 bits, lo que le permite leer dos instrucciones en cada acceso a memoria, y como se ve en el diagrama de bloques, dispone de un gran número de unidades funcionales, incluyendo dos módulos de procesamiento gráfico. Cada uno de estos módulos es independiente, lo que le permite ejecutar hasta 7 instrucciones de forma simultánea.

UN CLÁSICO, EL SPARC DE SUN

Hablar de tecnología RISC es hablar de Sun Microsystems; esta empresa fue pionera en el mundo de las estaciones de trabajo basadas en UNIX y, con la creación del Sparc, también sería pionera en el uso de microprocesadores RISC, la familia de procesadores que equipa sus ordenadores.

En RISC, todas las instrucciones tienen la misma longitud

Al igual que los Motorola 88000, en los Sparc las instrucciones también tienen una longitud de 32 bits; los formatos se pueden ver en la figura 8. La característica más destacada de estos procesadores es su juego de registros; los Sparc tienen entre 40 y 520 registros, aunque sólo pueden usar 32 de forma simultánea. Como puede verse en la figura 9, el juego de registros está dividido en dos grupos principales, el primero formado por los 8 registros estáticos, y el segundo formado por 24 registros situados en una ventana deslizante. En la llamada a una subrutina, el procesador hace avanzar la ventana y al retornar de la subrutina la hace retroceder; esto equivale a salvar el contenido de los registros de una forma similar, a como se hace con las instrucciones push y pop. Hay que decir que el desplazamiento no se hace de forma automática, se hace mediante las instrucciones save y restore. Como se ve en la figura, los desplazamientos son de 16 posiciones, lo que significa que los ocho registros que, antes de la llamada, ocupan las posiciones de la 24 a la 31 (posiciones 16 a 23 dentro de la ventana), tras la llamada pasan a ocupar las posiciones de la 8 a la 15 (de la 0 a la 7 dentro de la ventana); es decir, existe un solapamiento entre los registros de la rutina llamante y la llamada. Este solapamiento se utiliza para el paso de parámetros, así, las rutinas utilizarán los registros 24 a 31 para pasar parámetros a las rutinas que llaman y recibir resultados de estas, y los registros 8 a 15 servirán para recibir parámetros de las rutinas que la llaman, y para devolverle resultados; esto se puede ver en la figura 10. Este esquema permite definir cuatro grupos de ocho registros cada uno: los globales (comunes a todas las subrutinas), los registros OUT (para pasar parámetros a subrutinas), los locales (que no son accesibles desde subrutinas) y los IN (para recibir parámetros en las llamadas).

EVOLUCIÓN DE LOS RISC

Los RISC, como toda tecnología que se precie, sigue evolucionando; en particular, el ya mencionado 88110 es un ejemplo muy claro, del rumbo que ha tomado actualmente esta tecnología. Todos los RISC de última generación tienen buses de 64 bits, para poder leer las instrucciones de dos en dos, y su unidad de control incorpora multitud de módulos independientes, para poder ejecutar varias instrucciones de forma simultánea, pero esta capacidad superescalar presenta ciertos problemas, como por ejemplo, las instrucciones de salto; cuando se encuentra con una instrucción de este tipo, el procesador no puede seguir hasta que se ha terminado de ejecutar.

Para resolver estos problemas, se ha recurrido a procedimientos como la ejecución predictiva de saltos; el problema es que estas técnicas añaden nuevas formas de complejidad a la CPU, lo que obliga a preguntarse si a estos procesadores se les puede seguir llamando RISC. De todas formas, su existo comercial a medio plazo está garantizado. ■

HACIA UNA NUEVA OOP

Emilio Castellano

El Workframe de IBM ofrece un entorno unificado para desarrollar aplicaciones bajo OS/2, permitiendo configurar y adaptar a nuestras necesidades el entorno de desarrollo, así como personalizarlo con las herramientas preferidas de diseño (editores de código, gestor de iconos, editor de recursos, etc).

Una de las características mas interesantes del WorkFrame es que aprovecha el estándar gráfico de presentación WorkPlace Shell (sistema de presentación gráfico, o frontend de OS/2 introducido a partir de la versión 2.1). Este entorno gráfico, al que no todos los programadores C estarán acostumbrados, permite modelizar mediante objetos las tareas habituales de desarrollo, lo cual introduce una nueva filosofía de programación, muy en línea con las técnicas cada vez más frecuentes de modelización y programación visual.

Esta filosofía ayuda sobre todo a resolver uno de los grandes problemas a la hora de desarrollar una proyecto de programación de cierta envergadura.

Mediante los containers de información podemos agrupar el código en grupos lógicos, y obtener una visión más clara de nuestro proyecto.

Esto suele traducirse en una mejor y más estructurada programación, pues, el esfuerzo que dedicamos inicialmente a encontrar la estructura más adecuada para los módulos de nuestros programa, redundará posteriormente en una mejora en el código obtenido.

GESTIÓN DE PROYECTOS

Es destacable la posibilidad de trabajar con proyectos compuestos, que agrupan en su interior como secciones, otros proyectos. Por ejemplo el proyecto EJ1 podría contener a su vez : LIBPROY, DLLPROY, AYUDAPROY; para librerías, librerías dinámicas y ficheros de ayuda respectivamente.

ACCIONES

Este es también, uno de los conceptos novedosos en esta versión del kit de desarrollo. Cada objeto definido dentro del entorno, tiene asociada una serie de acciones, que podemos cambiar mediante la opción change action.

Esta función permite obtener una ventana flotante con las acciones definidas para ese objeto. (compilar, enlazar, editar, etc) al seleccionar uno de los objetos del proyecto, de la misma manera que nos aparece una ventana con opciones cuando seleccionamos algo en el Workplace Shell con el botón derecho del ratón.

GRUPOS DE TRABAJO EN RED Y WORKFRAME

Gracias a la posibilidad de instalación en red, el WorkFrame puede ser una herramienta muy poderosa en grupos de desarrollo, pues permite centralizar el desarrollo de determinados proyectos en un servidor, y distribuir el

Workframe de IBM ofrece un entorno unificado para desarrollar aplicaciones bajo OS/2

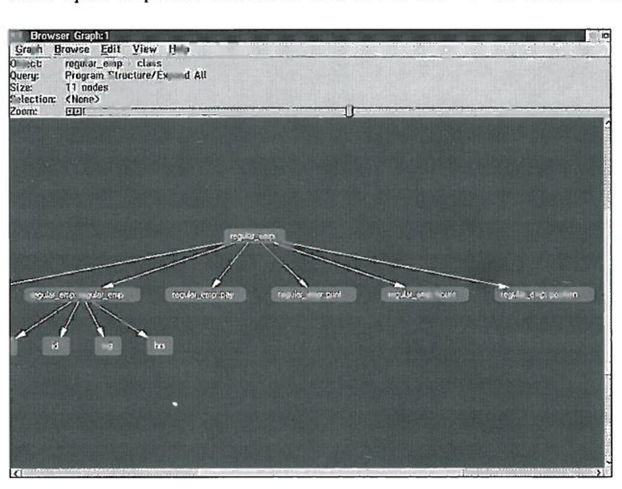
Al borde del nuevo siglo y cuando pensábamos, que no íbamos a recibir ya ninguna nueva sorpresa, desde la aparición de la programación orientada a objetos y los entornos de desarrollo visual, surge este entorno de programación en 32 bits, que promete acompañar al OS/2 en su metamorfosis hacia su nueva concepción para años venideros.

desarrollo a cada una de las estaciones de trabajo.

SOM

Podríamos definirlo como un estándar para definición de objetos bajo OS/2. Una forma de universalizar la declaración de nuestros objetos para así utilizarlos a nuestro antojo desde cualquier lenguaje que admita este estándar.

SOM incluye el OIDL (Object Interface Definition Language), un lenguaje para especificar el interfaz externo de una clase. El OIDL tampoco es un lenguaje de programación, no nos permitirá implementar una clase. Para ello deberemos utilizar un lenguaje C, C++, Smalltalk o cualquier otro que soporte enlaces SOM. Es importante comprender la relación existente entre SOM y el Workplace Shell. SOM es una tecnología de encapsulado, y el Workplace Shell es algo creado con esta tecnología.



C++ Browser: vista detallada de la clase *empleado*.

SOM es independiente del lenguaje, pues fue expresamente diseñado para poder expresar semánticas orientadas a objeto de muy diversos tipos.

Esto permite entre otras cosas variar la estructura de las clases, sin tener que cambiar las aplicaciones clientes de esas clases, así como evitar ciertos problemas que todo programador C++ se habrá encontrado en alguna ocasión, como por ejemplo, la imposibilidad de utilizar clases generadas con compiladores C++ diferentes, o tener que recompilar un programa si sustituimos el DLL del que depende.

UTILIDADES DE PROGRAMACIÓN DEL TOOLKIT

Mencionaremos aquí algunas de las utilidades más interesantes incluidas dentro del kit de desarrollo. Podemos diferenciar dos grupos dentro de las utilidades incluidas en el kit. Por un lado están las clásicas utilidades de "modo texto", que, al igual que en otros compiladores pueden ser ejecutadas desde la línea de comandos (aunque recomendamos se utilice la mayor parte desde el Workframe). Por otro lado tenemos también utilidades para el Presentation Manager, propias de todo entorno de programación orientado a gráficos (GUI) que se precie. Comenzaremos hablando de las primeras.

Mediante los containers de información podemos agrupar el código en grupos lógicos

EXEHDR: Permite obtener información sobre el número, y tamaño de segmentos de un programa, así como ver y cambiar otros atributos de un programa como los fijados por el módulo de definiciones.

FWDSTAMP: Añade puntos de entrada a una librería DLL, que indican referencias a funciones externas, ya sean de la API u otro código o datos exportados. Su función es poder combinar varias DLL en una sola, sin tener que recompilar una aplicación.

IMPLIB: Crea una librería de importación, a partir de un módulo de definiciones .DEF

KwikINF: Es un gestor para acceder a las ayudas online que incorpora el toolkit.

LINK386: Como su nombre indica es la utilidad de montaje incluida con el kit, que permite, no sólo generar ejecutables, sino también DLLs y controladores de dispositivos.

MKMSGF: Crea un fichero binario de mensajes, accesible a través de DosGetMessage, a partir de un fichero de texto. Esta función permite entre otras cosas tener mayor facilidad a la hora de mantener aplicaciones multilingües.

NMAKE: Al igual que en otros compiladores, el CSET++ también tiene un programa MAKE, que se encarga de la compilación selectiva de aquellas partes de un proyecto que ha sido modificadas desde la última ejecución de nmake.

Las utilidades Presentation Manager que citaremos son el IPFC (Information Presentation Facility Compiler), el Resource Compiler, el Dialog Editor, el Font Editor, y el Icon Editor.

De todos ellos, quizá el más interesante sea la primera, el IPFC, o compilador de informaciones en línea, que permite crear un sistema de ayudas en línea, del estilo de los incluidos en toda aplicación profesional.

El IPFC permite definir dos tipos de ayudas: documentos en línea, y ventanas de ayuda en los programas. Dentro del IPFC podemos diferenciar dos partes, un lenguaje que permite definir como y donde se mostrarán las ayudas en pantalla, y un compilador que convierte el fichero a formato IPF.

DEBUG KERNEL

Junto con el C SET se incluye una versión especial del kernel, el debug kernel, una versión especial del kernel de OS/2, que permite introducir puntos de ruptura y trazas, así como utilizar direcciones simbólicas.

Desafortunadamente esta versión del kernel es válida sólo para OS/2 2.1, por lo que los que estén programando bajo OS/2 WARP, deberán esperar a la versión 3 del kit de desarrollo.

C/C++ TOOLS

Dentro de la carpeta C/C++ Tools 2.01, encontramos lo que constituye realmente el núcleo del CSET++. Por un lado tenemos las clases, y por otro tenemos utilidades como el Class

Browser, el Execution Analizer, y el C/C++ Debugger.

C++ CLASS BROWSER

Esta utilidad nos permite analizar perfectamente la estructura de un proyecto C++. Gracias a él podemos listar los componentes del programa, al mismo tiempo que vemos una representación gráfica mediante grafos de las relaciones existentes entre las clases.

Mediante estos gráficos podemos, no sólo estudiar la herencia de propiedades entre unas clases y otras, sus correlaciones, y las llamadas a funciones, sino también editar el código asociado a un elemento del programa.

Si se desea utilizar está utilidad, será necesario compilar nuestro programa con la opción /Fb, que indica al compilador la necesidad de generar la información para el browser, un fichero .BRS en el que se describen los elementos del programa, y las relaciones de unos elementos con otros.

El browser provee de tres maneras diferentes de mostrar la información. El primer icono que encontraremos al arrancar el browser es list, que nos da una lista de los elementos incluidos en el proyecto, código fuente, ficheros, funciones y clases. También tenemos el icono text que nos muestra el código del elemento que tengamos seleccionado, y nos permite editarlo.

WorkFrame puede ser una herramienta muy poderosa en grupos de desarrollo

Por último tenemos graph que es el elemento fundamental del browser. Desde aquí podemos ver el grafo de uso y las relaciones entre todos los componentes del proyecto, así como realizar todo tipo de búsquedas y consultas (queries). Las relaciones de uso entre los nodos están indicadas mediante líneas, y se utiliza un código de colores o diferentes estilos de líneas para especificar el tipo de relación. Las relaciones cíclicas se indican en el grafo mediante arcos, que apuntan al nodo de retorno. Una vez que nos encontramos en el grafo podemos comprimir, o expandir la estructura de forma que podamos ver mejor la precedencia entre clases.

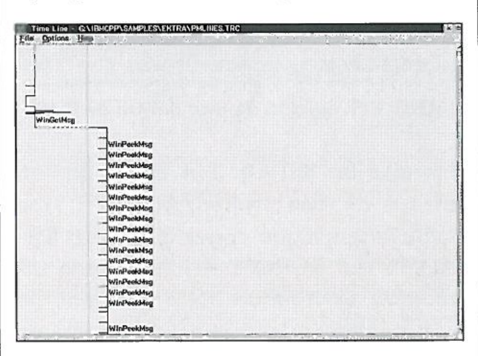
```
Disassembly: MAHJONGG:1 Thread 1
File  Edit  Breakpoints  Windows  Run  Options  Help
:00013EFC  LEAVE
:00013EFD  RET
:00013EFE  XCHG   EBX,EBX
:00013F00  PUSH   EBP
:00013F01  MOV    EBP,ESP
:00013F03  FINIT
:00013F05  WAIT
:00013F06  FLDCW  [000217CCH]
:00013F0C  PUSH   00014FC8H
:00013F11  PUSH   DWORD_FS:[00000000H]
:00013F18  CALL   +000000CBH      [00013FE8]
:00013F1D  CMP    EAX,-00000001H
:00013F20  JZ     +6CH            [00013F8E]
:00013F22  LEA    EAX,[-8H+EBP]
:00013F25  MOV    FS:[00000000H],EAX
:00013F2B  MOV    EAX,0001A258H
:00013F30  SUB    ESP,00000004H
:00013F33  CALL   +000017D8H      [00015710]
:00013F38  ADD    ESP,00000004H
:00013F3B  MOV    EAX,[+14H+EBP]
:00013F3E  SUB    ESP,00000004H
:00013F41  CALL   +00000382H      [000142C8]
:00013F46  PUSH   00013ED8H
:00013F4B  MOV    EAX,0000FF01H
:00013F50  PUSH   EAX
:00013F51  MOV    AL,02H
:00013F53  CALL   +1BF7C138H      [1BF90090]
:00013F58  ADD    ESP,0000000CH
:00013F5B  CALL   +00000648H      [000145A8]
:00013F60  MOV    ECX,[0002426CH]
:00013F66  PUSH   ECX
:00013F67  MOV    EDX,[000217C8H]
```

C/C++ Debugger: código desensamblado por el C/C++ debugger.

EXECUTION TRACE ANALYZER

El analizador de trazas de ejecución, habitualmente denominado EXTRA, es una aplicación OS/2 que permite analizar y optimizar los programas generados en este sistema. EXTRA permite trazar la ejecución de un programa, generando un fichero con las trazas y un conjunto de elementos de análisis, que pueden después ser mostrados con todo tipo de diagramas.

A través de EXTRA se puede por ejemplo obtener un registro detallado del momento en que se produjo cada evento, el tiempo acumulado por cada procedimiento y muchos otros datos. Esto permite estudiar los cuellos de botella de nuestros programas, puntos en los que nuestra aplicación dedica un tiempo excesivo.

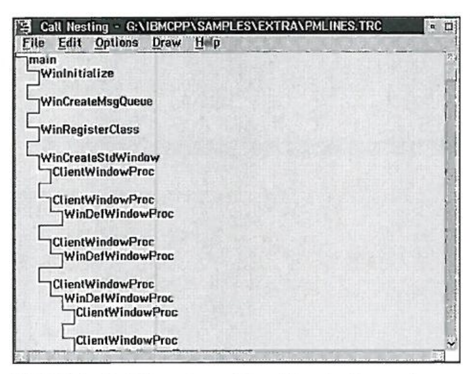


EXTRA: Opción de visualización mediante gráfica de tiempos.

Esta utilidad también puede sernos de gran ayuda, si tenemos un programa que sufre de deadlocks (bloqueos), es decir momentos en los cuales nuestra aplicación se queda bloqueada. Esta es precisamente una de las funciones más interesantes de esta utilidad, si se programan aplicaciones multitarea, pues debido a las relaciones entre diferentes eventos y recursos, pueden darse situaciones en las que el bloqueo sea casi aleatorio, las cuales sería casi imposible aislar y corregir, si no contásemos con una herramienta de estas características.

C/C++ DEBUGGER

El debugger, como habitualmente se le conoce, es una aplicación Presentation Manager, que permite detectar y diagnosticar errores en aplicaciones C/C++ de 32 bits. Un punto novedoso de esta versión, es que permite trabajar con aplicaciones C++, contando además con un conjunto de funciones específicas. Por ejemplo, esta versión del debugger tiene la posibilidad de depurar funciones template, muestra las herencias entre clases de forma



EXTRA: Gráfica de anidación de llamadas.

gráfica, permite evaluación de expresiones C++, así como extraer información detallada sobre una clase.

Básicamente el debugger tiene tres vistas en las cuales se muestra el programa que estamos depurando, pudiendo establecer puntos de ruptura, examinar variables y registros, o analizar el estado de la pila.

CLASES C++

El kit de programación que nos ocupa, agrupa un conjunto de 260 clases y más de 2600 funciones. Estas están reunidas básicamente en los siguientes grupos:

- Clases generales: para gestión de procesos, recursos, etc.
- Tipos de datos: los tipos de datos string, points, etc están contenidas en clases.
- Gestión de eventos: para controlar las acciones de los usuarios, o intervención de otras aplicaciones.
- Gestión de ventanas: permiten definir y controlar, todos los elementos gráficos de nuestra aplicación, desde simples ventanas o cajas de diálogo, hasta elementos compuestos.

CSET++ BAJO WARP

El kit de desarrollo 2.1 puede ser instalado bajo OS/2 WARP, no obstante, debido a los cambios introducidos en esta versión del sistema operativo, IBM ha publicado una lista de "parches" para aplicar al compilador, a las utilidades, y a las librerías de clases para adaptar el kit al funcionamiento bajo Warp. Algunos de los parches son recomendables, pero otros son imprescindibles.

Están disponibles en la BBS de IBM en los teléfonos: 3975963 - 3975873 - 3975581 - 3975580.

Por poner un ejemplo, uno de los fallos que podemos obtener, se consigue si se utilizan las funciones getcwd, o getdcwd. Estas llamadas producen un error si no se le pasa un buffer en la llamada.

INSTALACIÓN

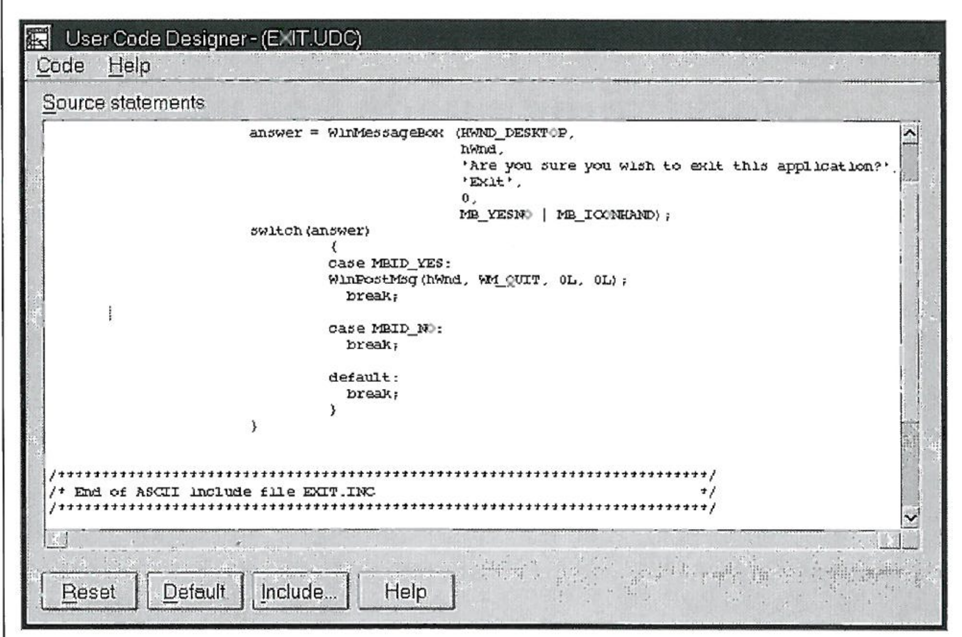
Este es uno de los puntos flojos del paquete de desarrollo para OS/2. La versión analizada, (la distribuida sobre CDROM), no posee un único programa que permita simplificar la instalación del paquete.

SOM es independiente del lenguaje y permite variar la estructura de las clases, sin tener que cambiar las aplicaciones clientes

Como es de suponer, aquél que se decide a comenzar a desarrollar bajo OS/2 no suele ser un programador novel, y será capaz de llevar a cabo la instalación del kit. Sin embargo, no estaría de más un INSTALL.EXE, que evitase tener que coger el fichero README del CDROM, e ir por cada directorio ejecutando los programas de instalación individuales de cada una de las secciones (Toolkit, C/C++ tools, WorkFrame, etc). Un ejemplo más de la falta de unidad en el proceso de instalación, la encontramos en el hecho de tener que instalar el WorkFrame en dos partes, primero el WorkFrame 1.1, y después encima de éste la versión 2.1.

KASE:SET

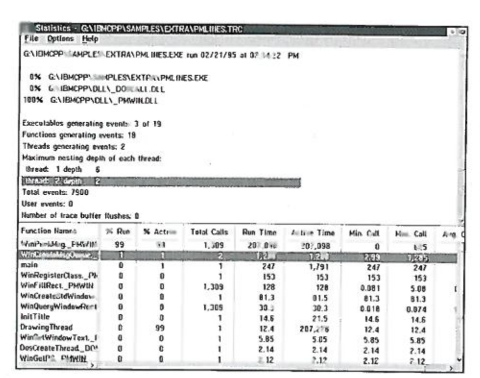
Procedente de la casa de software KASEWORKS, KASE:Set introduce el concepto de la programación visual en este entorno de programación 32 bits. KASE: Set es una versión reducida del paquete KASE: Vip, especialmente diseñado para el entorno WorkFrame de IBM, que se integra perfectamente en este último, y nos permite diseñar nuestra aplicación gráficamente y generar código C/C++.



KASE: Set, edición de uno dentro de KASE: Set.

Gracias a este paquete se puede por ejemplo diseñar el aspecto de la aplicación, sus menús, paneles de mensajes, aparte de permitirnos realizar un seguimiento del proceso de desarrollo.

La versión completa KASE: Vip (Visual Integration Platform, plataforma de integración visual), permite otras



EXTRA: Estadísticas de llamadas a las funciones.

funciones más potentes, como por ejemplo:

- Control de librerías de enlace dinámico.
- Sentencias SQL
- Interfaz con el sistema de ayudas del Presentation Manager
- Un entorno IDE que permite organizar en carpetas los objetos de desarrollo, nuestras utilidades de desarrollo.

CONCLUSIÓN

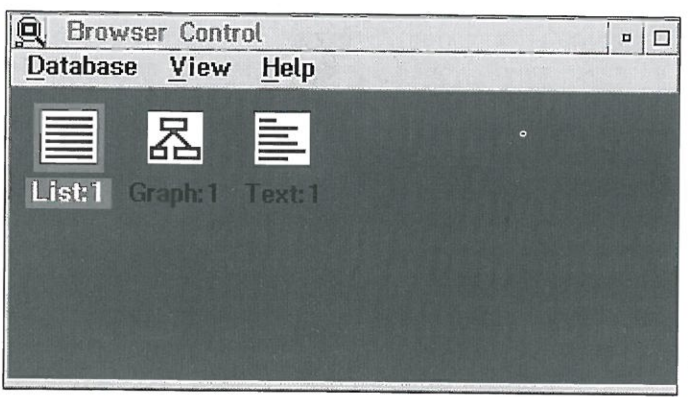
El C SET/2 2.1 es una herramienta potente de programación 32 bits, pero lo más interesante en este kit de programación es la inclusión de la tecnología SOM.

La tecnología OOP actual tiene la limitación de la especialización de las librerías de clases en cada lenguaje. Una librería C++, no puede ser utilizada por ejemplo, bajo Smaltalk, o Pascal ++ y viceversa. Es en este aspecto, donde la tecnología SOM puede significar una revolución, y es que SOM provee al programador de un modelo de programación independiente del lenguaje.

La encapsulación que ofrece, permite esconder los detalles de la implementación de las clases de los clientes de las mismas. Es decir, que un cambio en la estructura de una clase no afectará a los programas clientes de la misma, y una mala programación de uno de ellos no afectará a nuestras clases. Además, SOM permite también herencia y polimorfismo.

El IPFC nos permite crear un sistema de ayudas en línea, del estilo de los incluidos en toda aplicación profesional

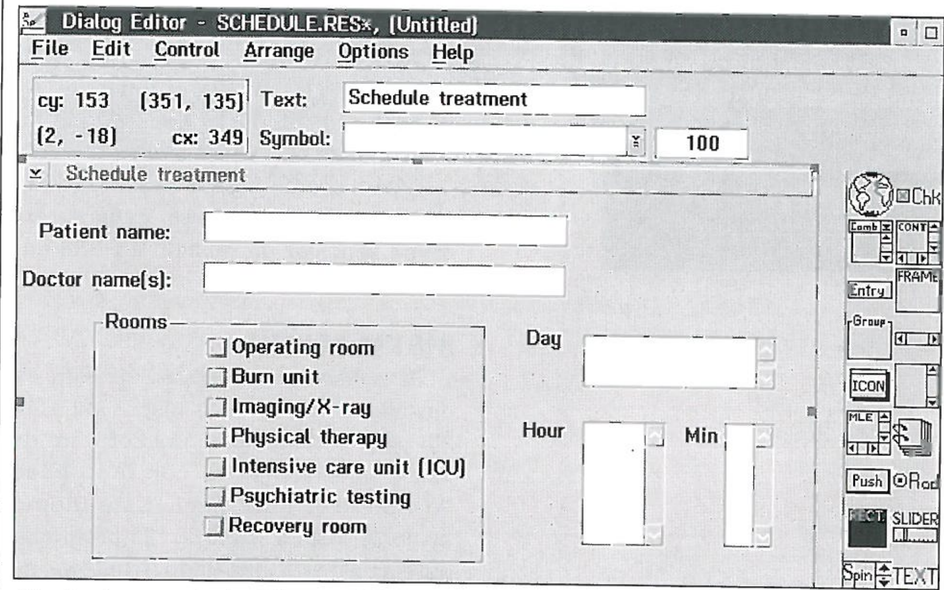
El entorno de programación del C SET/2, a pesar de ser muy potente, presenta quizá la pega de no resultar familiar para quien se encuentre acostumbrado a entornos de desarrollo más amigables, "tipo Borland", no obstante, el premio que se encuentra detrás de esta primera dificultad, bien merece la pena el esfuerzo de adaptación.



C++ Browser: las tres opciones del browser.

DISTRIBUCIÓN

El C SET/2 podemos encontrarlo en muchos formatos, cada uno con un precio diferente, cada usuario interesado en el producto debe decidir cual se ajusta mejor a sus necesidades. Las tres presentaciones contienen la misma información, sólo cambian los costes de duplicación y presentación, que son en definitiva los que marcan el precio final de los paquetes:



Diseño de una caja de diálogo en KASE: Set.

Si lo desea puede adquirir una versión reducida del C SET/2 denominada C SET/2 First Step, que incluye el compilador de 32 bits con sus librerías, el debugger, el WorkFrame 1.1, y el kit de desarrollo, cuyos precios son:

C SET/2	
en CD	71.940
en CD + manual	78.390
en disquetes de 3 1/2"	83.760
C SET/2 FirstStep	
en CD	15.900
en CD + manual	18.260
en disquetes de 3 1/2"	18.260
Toolikit para Warp	5.100

A estos precios de venta al público, hay que añadirles el IVA correspondiente. ■

LANtastic 6.0

Carlos Arias

Lantastic 6.0 es una red local peer-to-peer para DOS y WINDOWS con avanzadas prestaciones de seguridad, correo electrónico, fax, buscapersonas, etc. Soporta entornos mixtos con posibilidad de conexión a servidores NETWARE, UNIX y OS/2. Comparte impresoras, CD-ROM, unidades WORM y unidades no DOS. Es un producto consolidado en el ámbito de las redes locales.

Está especialmente recomendado para el entorno Windows, dada su extrema sencillez de manejo en este entorno.

LANtastic 6.0 es una solución en aquellos sitios donde se necesita conectar PCs para compartir programas, ficheros, impresoras, unidades de CD-ROM, Modem/Fax o intecambiar mensajes.

INSTALACIÓN

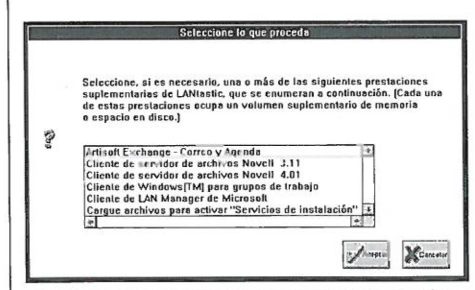
El software de instalación está en castellano. Viene en 4 discos de 1'44 Mb para la instalación del software de red y un disco extra con la utilidad WordPerfect FaxDirect. Este último disco se instala a parte. El programa detecta si está instalado Windows en el ordenador, y en caso afirmativo lanza la instalación para este entorno. Lo primero que pide es el nombre que se va a dar al ordenador dentro de la red y la unidad y directorio donde se van a copiar los ficheros, que por defecto es la unidad C y el directorio LANTASTI. A continuación, pregunta si se desean compartir las unidades e impresoras. Si se responde afirmativamente el ordenador se configura en modo servidor y pregunta por el número máximo de estaciones soportadas, que van de 10 (la configuración por defecto) a un total de 100.

Si queremos que el ordenador se conecte a las impresoras y unidades de un servidor cuando se arranque, hay que responder afirmativamente a la pregunta de instalar dispositivos permanentes. También se puede indicar que no se conecte y realizar la conexión manualmente mas adelante.

Dependiendo de las necesidades del ordenador se pueden instalar las siguientes utilidades adicionales:

- Artisoft Exchange: Agenda, correo electrónico, envío de fax, gestión de mensajes, buscapersonas, etc. Sólo está disponible para Windows.
- Cliente de servidor de archivos Novell Netware 3.11 y 4.01, Windows para Trabajo en grupo y Lan Manager de Microsoft..
- Servicios de instalación remota.

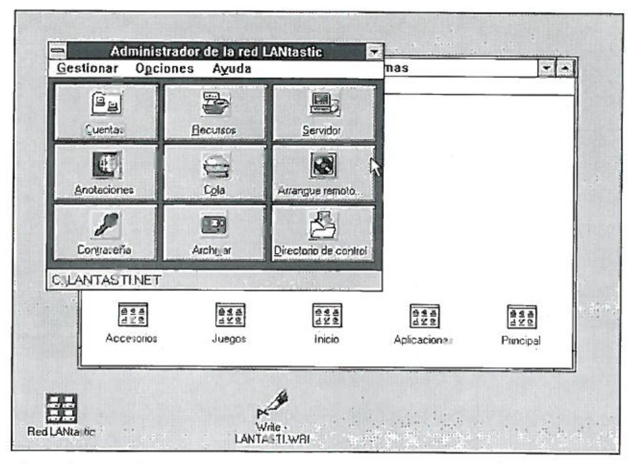
Si se incluye la utilidad de correo Artisoft Exchange hay que indicar el tipo de oficina postal a utilizar. Puede ser cliente de correo (se copian los programas de correo en el ordenador),



Accesorios cargables durante la instalación.

cliente de correo en red (no se copian los programas de correo, ya que estos se ejecutan desde la oficina de correo), y gestor de correo electrónico, que hace de servidor de correo. Debe de configurarse al menos un ordenador de la red como oficina postal). A la oficina postal se le tiene que asignar un nombre. Se pueden utilizar las mismas cuentas o cuentas separadas para la red Lantastic y el correo Artisoft Exchange.

Después de realizar toda esta serie de preguntas pide la confirmación de las opciones, para en caso de habernos equivocado, repetir el proceso.



Administrador de la red Lantastic.

Los fichero que modifica son el AUTOEXEC.BAT, CONFIG.SYS, SYSTEM.INI, WIN.INI y PROGMAN.INI. Las antiguas versiones de estos ficheros los guarda en otros con el mismo nombre pero con extensión ".001". Es un detalle, que muestre al usuario los nombres de los ficheros del sistema que modifica, así como las extensiones que da, a la versión antigua de éstos, no como otros programas que los modifican sin informar para nada al usuario.

Se pueden mezclar equipos basados en MS-DOS y en Windows

Cuando se ha finalizado la instalación, Lantastic crea una nueva carpeta llamada LANtastic en el administrador de programas conteniendo nueve iconos.

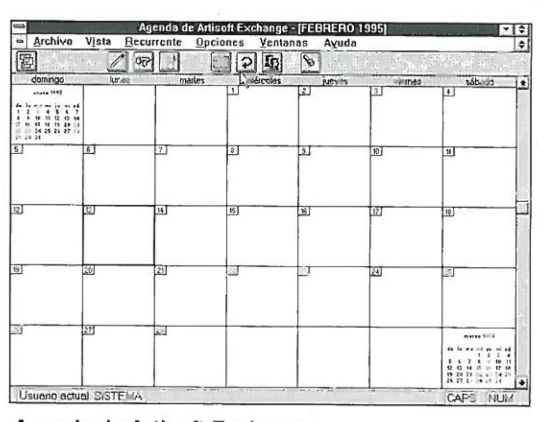
La tarjeta de red que se suministra es la NodeRunner /SI 2000/C de la misma casa. No lleva jumpers para configuración por hardware, ya que ésta se realiza por software. Viene de fábrica configurada con la IRQ 3 y la dirección de puerto 300h. Se puede instalar en ranuras de 8 y 16 bytes, pero sus máximas prestaciones las obtiene en ranuras de 16 bits. Es compatible con Novell Netware, Microsoft Windows WorkGroups, Banyan VINES y otras redes ethernet. Dispone de zócalos para ROM de arranque, útil en equipos desprovistos de disco duro y disquetera. Se entrega con un folleto en castellano de fácil comprensión.

El soporte para tarjetas de red de otras marcas es bastante amplio, pudiendo utilizar más de 27 tipos de tarjetas distintas. Las pruebas realizadas en este examen se realizaron primero con las dos tarjetas propias que trae el kit de instalación, y con tarjetas compatibles Novell NE2000. Para indicar la dirección del puerto y la interrupción por hardware de la tarjeta hay que variar el fichero STARTNET.BAT y el fichero SYSTEM.INI, y buscar en este último el apartado de la sección [Lantastic].

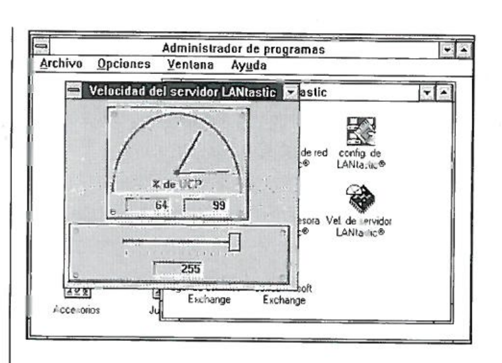
Como último detalle, trae la posibilidad de poder realizar la actualización de la versión de la red desde un servidor.

FUNCIONAMIENTO

Lo primero que sorprende al entrar en Windows es la baja cantidad de recursos de memoria que utiliza este paquete para gestionar la red. Nada más que 70 Kb frente a los 700 que utiliza Windows para trabajo en grupo. Es por esta razón por la cual resulta muy conveniente su instalación, aun cuando se posea Windows para trabajo en grupo, sobre todo en ordenadores equipados con 4 Mb de memoria RAM y una velocidad de proceso no muy elevada. La interfaz de trabajo para las labores comunes de conexión a recursos compartidos es muy amigable, basándose en la tecnología de "arrastrar y soltar".



Agenda de Artisoft Exchange.



Si la seguridad de la red es importante, lo primero que se ha de crear para el acceso a los recursos son las cuentas de usuario. El mantenimiento se lleva a cabo desde el icono Administrador de la red Lantastic en el grupo de programas Lantastic. Se pueden utilizar cuentas individuales o de grupo, éstas últimas llamadas en esta nueva versión cuentas comodín, para no confundirlas con los grupos LCA (Lista de Control de Acceso).

CONFIGURACIÓN DE LAS CUENTAS DE USUARIO

Como se ha indicado anteriormente las cuentas pueden ser individuales o comodín. En ambas hay que indicar un nombre, una contraseña, si se quiere, y el número de conexiones simultáneas soportadas para esa cuenta. Si en una cuenta individual se indica más de una conexión, el usuario podrá acceder al mismo tiempo desde diversos nodos. Una vez configuradas las cuentas para un servidor, éstas pueden ser copiadas a otros servidores, lo cual permite configurar la red de una manera más rápida y eficiente.

SEGURIDAD DE LA RED

Posee varios niveles de seguridad. Además de pedir una contraseña, se puede indicar los días y las horas que una determinada cuenta de usuario puede acceder a los recursos del servidor. Se puede forzar a que el usuario cambie su contraseña cada cierto tiempo. También se puede indicar una fecha de expiración automática de la cuenta, opción que viene muy bien para dar acceso a cuentas visitantes sin preocuparse de tener que darlas

de baja de forma manual. Por último, se pueden activar privilegios adicionales que permiten al usuario disponer de tipos opcionales de acceso. Estos privilegios especiales otorgan a la cuenta la posibilidad de controlar la administración de los recursos de la red, controlar los trabajos de la cola de impresión a nivel global, gestionar los mensajes de una oficina postal, para retirar el correo antiguo, crear anotaciones en el registro de auditoría del servidor, estadísticas de la red, etc.

UTILIZACIÓN DE RECURSOS COMPARTIDOS

Se lleva a cabo a través de la opción unidades de la utilidad Conexiones de Red Lantastic. Ésta se carga automáticamente cuando se arranca Windows y permanece minimizada hasta que se la invoca. El método para "capturar" un recurso del servidor no puede ser más sencillo. Se toma el servidor, para que todos los usuarios tengan acceso a él. Ahorra tiempo en la actualización y mantenimiento de las aplicaciones. Se puede poner a disposición de la red un disco entero, o solamente los directorios que interese. Se pueden configurar las impresoras para que funcionen en distintos modos de impresión. Si se poseen unidades CD-ROM o discos WORM, se pueden compartir con el resto de la red. El servidor se puede conectar a discos NetWare, así como a discos OS/2 HPFS, poniendo a disposición de la red dicho recurso.

GRUPOS DE LISTA DE CONTROL DE ACCESOS (LCA)

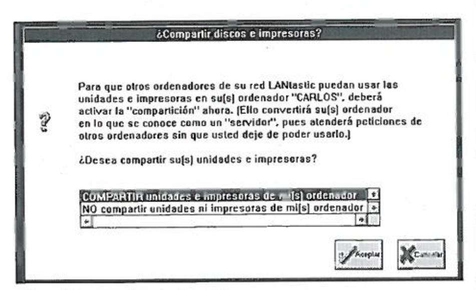
Mediante esta opción , se especifica el tipo de acceso que un usuario, o un grupo de usuarios tiene a un determinado recurso. El nivel de acceso a los recursos no se definen en la propia cuenta de usuario, sino mediante la asociación de las plantillas LCA a las cuentas.



La instalación de LANtastic es muy sencilla.

El paquete incluye todo lo necesario para instalar la red

recurso que se desea utilizar de la ventana izquierda, donde se muestran de manera gráfica todos los recursos del servidor disponibles, y se arrastra con el ratón a la ventana derecha, donde se muestran las letras de las unidades disponibles para poder asignar el recurso. Se utiliza el mismo procedimiento para conectar impresoras, CD-ROM o tomar el control remoto de la pantalla y el teclado.



Opción de instalación.

El nivel de gestión de los recursos compartidos es bastante alto. Permite ejecutar a nodos de la red, programas que se encuentren instalados en el servidor. De esta manera, sólo hace falta cargar una copia del programa en el

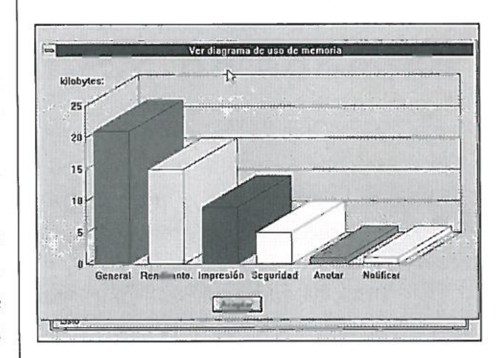
A las plantillas LCA creadas por este sistema, se les puede asociar cuentas individuales o comodín, que vayan a compartir las mismas características de acceso. De esta forma se gana velocidad, a la hora de configurar el acceso a los recursos de la red.

Se pueden definir niveles de acceso para una unidad de disco, un directorio o un único fichero. Los niveles de acceso a los recursos de la red son los siguientes:

- Acceso de lectura
- Acceso de escritura
- Creación de archivos y directorios
- mostrar archivos y directorios
- borrar archivos y directorios
- renombrar archivos
- ejecución de programas
- cambio de atributos.

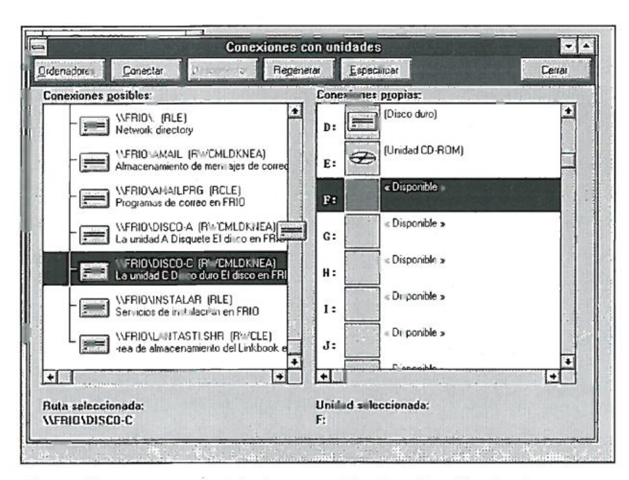
UTILIZACIÓN DE RECURSOS GLOBALES

Los recursos globales son definidos en un servidor como propios, pero realmente no pertenecen a él, ya que señalan a recursos existentes en otro servidor. De esta manera, el usuario accede a un único servidor para utilizar los recursos de varios servidores, sin preocuparse de donde reside realmente el recurso. Por ejemplo, se pueden definir recursos globales pertenecientes a un servidor NetWare, creando un entorno mixto y centralizando todos los recursos en un único servidor. Esta técnica es similar a los recursos de dominio existentes en otros sistemas de red local. Hay que tener en cuenta, que esta técnica no ralentiza la comunicación del cliente con el recurso global, dado que el servidor lo único que hace es gestionar la conexión, no haciendo de intermediario en la transferencia de información entre las estación de trabajo y el servidor, propietario del recurso.



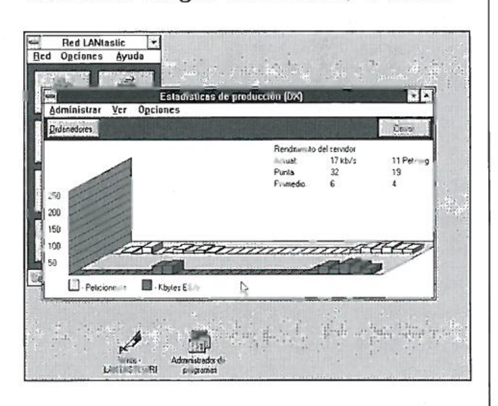
CONTROL REMOTO DEL SERVIDOR

Se puede tomar el control del teclado y el monitor de un servidor para realizar operaciones en él desde cualquier nodo de la red. El único requisito necesario, es que el servidor sobre el que se tome el control no esté funcionando en modo gráfico. También se pueden enviar ficheros batch al recurso de memoria de teclado del servidor, indicando la fecha y hora en que se desea que se ejecute. De esta manera, es como si se tecleasen directamente las órdenes en el servidor, en la fecha y hora indicada.


Conexiones con unidades, mediante el método de arrastrar y soltar.

CONFIGURACIÓN AVANZADA DEL SERVIDOR

La configuración avanzada del servidor es igual de fácil de utilizar, que los recursos compartidos. Cuando se accede a ella aparece una ventana con dos divisiones, en la parte izquierda aparecen los módulos de configuración disponibles, y en la parte derecha los módulos cargados en memoria, junto con la memoria que consume cada uno. Para cargar un módulo, o descargarlo de la memoria, se arrastra con el ratón al lado de la ventana deseado. Por ejemplo, si no se va a utilizar el ordenador como servidor de archivos, se coge con el ratón el icono que representa al módulo de impresoras, y se arrastra al lado izquierdo para descargarlo de la memoria. De esta manera se deja memoria libre para optimizar las funciones del servidor. Para configurar uno de los módulos, se hace doble click sobre el icono que lo representa, mostrándose a continuación la información detallada sobre sus parámetros, permitiendo su ajuste.



Se puede configurar la velocidad del servidor, de tal manera que atienda con mayor frecuencia a las peticiones de estaciones remotas, o por el contrario para que dé mayor prioridad a las tareas locales, si no se utiliza como servidor dedicado. El ajuste depende de la utilización que se vaya a dar al ordenador. Si se va a utilizar como servidor dedicado, conviene aumentar la velocidad al máximo valor posible. Una ayuda para ajustar mejor la configuración es consultar las estadísticas de producción y rendimiento del servidor, representadas por gráficos de barras, que indican la carga de la red local. Antes de realizar un ajuste sobre cualquier parámetro de rendimiento, conviene examinar estas gráficas. Si se desea aumentar la velocidad de las tareas de impresión, se puede asignar al buffer que se dedica a la cola de impresora mayor cantidad de memoria.

LANtastic puede configurarse como red de servidor, de igual a igual o mixta

DIÁLOGO ENTRE ORDENADORES

Se puede establecer un diálogo full-duplex entre dos usuarios de la red. Cuando se accede a esta opción, se muestra una ventana con dos áreas de edición, en las cuales se muestran los mensajes que los usuarios van introduciendo a través del teclado. No se puede decir que sea una utilidad muy elaborada, pero cumple con su cometido.

ARTISOFT EXCHANGE

Artisoft Exchange es la aportación realizada por Artisoft a los grupos de trabajo. Permite enviar, recibir e imprimir correo electrónico. Posee un corrector ortográfico incorporado, para su utilización con los mensajes, permite enviar y recibir faxes, enviar mensajes y números de teléfono a un "busca", y programar citas. En MS-DOS sólo se puede instalar la oficina postal. Vamos a ver a continuación las opciones más destacadas.

Envío de mensajes

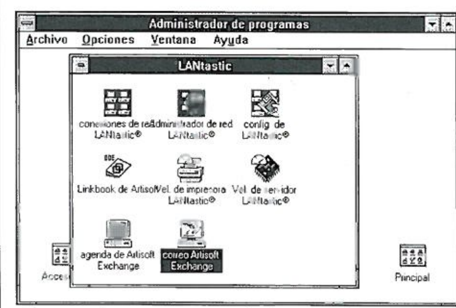
Envía mensajes a otros nodos de la red. Destacan las opciones de corrección ortográfica, cambio de la fuente de letra, de color, etc. Permite definir la prioridad entre urgente, normal, y baja. Se puede indicar si se pide un acuse de recibo, así como la posibilidad de adjuntar con el mensaje un fichero de datos, ejecutable, de fax o de voz, para escuchar con una tarjeta de sonido.

Envío de fax

Permite utilizar una tarjeta módem/fax conectada a un servidor para el envío de fax desde nodos de la red local. Se puede configurar la tarjeta módem/fax como si se tratase de una impresora, para la utilización desde aplicaciones externas a Artisoft Exchange.

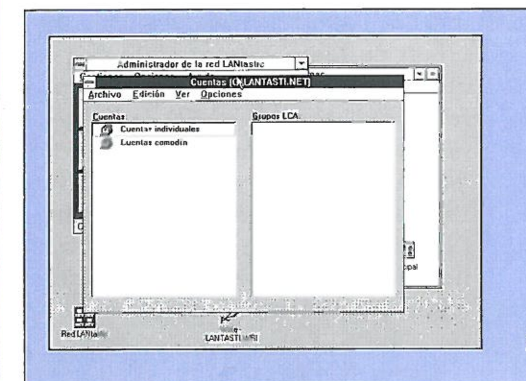
Buscapersonas

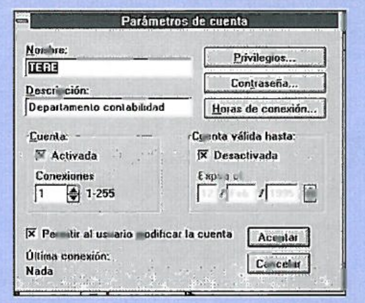
Permite enviar un número de teléfono a través de un módem. Si se utiliza el buscapersonas alfanumérico, se puede escribir un mensaje de 256 caracteres como máximo.

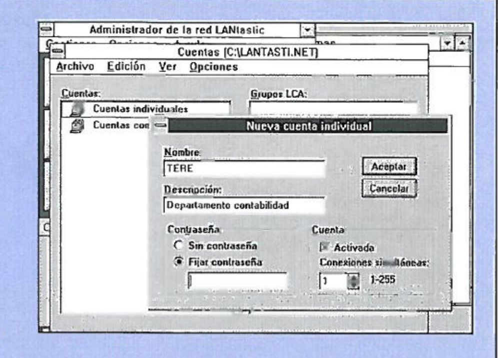

Grupo de programas de Lantastic.

Agenda

Permite configurar citas personales y de grupo, coordinando la disponibilidad de horas de los distintos miembros del grupo.







Parámetros de las cuentas del servidor.

OTROS DATOS DE INTERÉS

ALONE

Con la utilidad ALONE se puede utilizar un ordenador como servidor dedicado, aumentando las prestaciones. En el servidor dedicado se muestra en la pantalla las peticiones que realizan los nodos de la red.

UPS

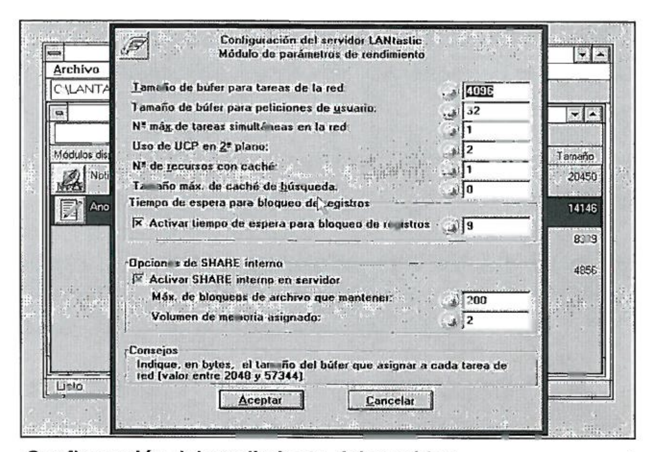
Permite a un servidor monitorizar un sistema de alimentación ininterrumpida, programando un apagado antes del agotamiento de las baterías.

LANUP

Es una utilidad residente para MSDOS, que permite gestionar las conexiones del ordenador. Se carga en memoria y se activa mediante una combinación de teclas.

RPS

Es un servidor de impresora remota, que permite reenviar los trabajos de la cola de impresión de un servidor a otras impresoras.



Configuración del rendimiento del servidor.

VALORACIÓN FINAL

Lantastic es un producto muy cuidado, provisto de una interfaz muy intuitiva. Gracias a la tecnología que incorpora de "coger y arrastrar" su manejo es bastante fácil. En cuanto a la seguridad, hay que destacar las numerosas opciones que incorpora, de restricción de acceso a recursos, llegando al nivel de poder restringir el acceso a unas horas determinadas, además de los niveles de seguridad mínimos exigibles a toda red local.

Permite además, conectarse a unidades no DOS, como discos duros NetWare, y OS/2 HPFS, ampliando las posibilidades de la red. En definitiva, es un producto a tener muy en cuenta sobre todo si se piensa instalar una red mixta, formada por equipos funcionando bajo MSDOS, WINDOWS y no se quiere perder la posibilidad de conectarse a servidores NetWare, Lan Manager y otros. ■

Es posible conectar la red a otras estaciones de tipo OS/2 o Unix

Panel de control del servidor Lantastic.

CONTENIDO DEL PAQUETE

2 tarjetas de red NodeRunner/SI 2000/C

Cable Ethernet fino 10 Base2 RG58A/U

2 terminadores.

2 conectores en T.

7 metros de cable.

5 disquetes de 3.5 con el software de la red y la utilidad WordPerfect FaxDirect

3 manuales en castellano:
- Guía de instalación y manejo de Lantastic.
- Guía básica al trabajo cotidiano con Lantastic.
- Instrucciones de uso de la agenda y el correo Lantastic.

1 plantilla de guía rápida.

FICHA TÉCNICA:



80286 o superior.
MSDOS, Windows 3.1.
4Mb RAM.
Distribuidor: CENTROID
Teléfono: 739 90 11
P.V.P.: 80.390 + IVA



COMO Y DONDE HACER FTP

Francisco G. Avilés

En este artículo se pretende explicar y guiar al usuario de Internet en el manejo del FTP (File Transfer Protocol), como uno de los servicios mas importantes, ya que hay en la red un gran número de ordenadores "Servidores de FTP anónimos (anonymous)" que ponen a nuestra disposición una enorme cantidad de programas de dominio público/shareware y todo tipo de información científico-técnica, normalmente generada por estudiantes, profesores de Universidades e investigadores y en algunos casos programadores de grandes empresas.

En la red Internet hay varios servicios o recursos, entre los mas conocidos están el E-Mail o correo electrónico para el envío o recepción de mensajes,el servicio Finger para consulta de información sobre un usuario, el FTP anónimo para acceso público a información en servidores de archivos, las listas de correo, los Archie, los WWW, los Gopher, los IRC ,etc. Su significado y complejidad se puede ver en el fichero INT-FAQ.TXT del disco de SÓLO PROGRAMADORES.

En la mayoría de las BBS no se tiene acceso directo a ftp, pero puede hacerse indirectamente este servicio a través del correo electrónico, al conectar a unos servidores llamados ftpmail. Igualmente con el indicativo EMAIL3 en ibertext llamando al 033 se puede, con el correo electrónico incluir los comandos necesarios para realizar FTP anónimo, aunque no lo recomendamos por coste y por la falta de operatividad del videotexto. Para practicar, en principio puede valer, al ser público y accesible a cualquiera con solo pagar a Telefónica. Véase la figura 1 de Ibertext.

Con el correo electrónico habrá que enviar un mensaje a un servidor ftpmail y dentro del mensaje colocar las órdenes específicas con comandos que vayan conduciendo la sesión FTP deseada. Estos comandos que nos recuerdan el sistema operativo UNIX, se pueden ver en la figura 2, Comandos del FTPmail.

Si tenemos acceso ftp normal, recordemos que la dirección, llamadas IP, se compone de números menores de 255 separados por puntos, o sea XXX.XXX.XXX.XXX, por ejemplo 128.102.18.3 corresponde a un ordenador de la NASA abierto al público. En algunos sistemas nos basta con utilizar pseudónimos por ejemplo, "ames.arc.nasa.gov" y el servidor internet al que estemos conectados lo traducirá a su dirección IP.

Esta última denominación es la llamada FQDN o Full Qualified Domain Name (nombre completo de dominio), que creemos es mas interesante, y así es mejor decir redes1.unizar.es para referirse al nodo de la EUITIZ que su dirección IP (155.210.23.10). En el fichero ESPAÑA.TXT incluido en el disquete de este mes pueden verse direcciones de ftp en España.

CONEXION CON FTP

Una sesion normal, que podríamos realizar en las facultades, escuelas o centros con conexión a Internet podría ser:

$ftp ftp.xx.es

Connected to ftp.xx.es.

Al pedirnos el nombre, escribiremos anonymous. A continuación nos pide la palabra clave o password donde introduciremos nuestra identificación con el siguiente formato: usuario@mi-ordenador.dominio.es. Normalmente nos deja-

Desde 1984, cuando Truevision definió un nuevo formato gráfico para el uso de los primeros productos videográficos, el formato TGA se ha mantenido como el más utilizado en ambientes profesionales, comprime poco, pero su calidad y facilidad de uso lo han mantenido totalmente vigente.

DÓNDE PODEMOS HACER FTP ANÓNIMO

La cantidad de servidores de ftp crece dia a dia en todo el mundo, a cual más interesante para programadores, y usuarios de informática.
Aquí se señalan algunos de los más interesantes:

-FTP sobre 3D

Dirección:	callamer.com (129.65.16.125)
Directorio:	/xing
Contenido:	MPEG para DOS o WINDOWS.
Dirección:	csg.lbl.gov (128.3.112.84)
Directorio:	/pub/listserv/photo-3d
Contenido:	3D FAQ, información de lentes, filtros, incluso versión 3D de Tetris.
Dirección:	exaic.wu-wien.ac.at (??.??.??.??)
Directorio:	/x11/games/spacial
Contenido:	Juegos 3D (X windows, Tetris 3D)
Dirección:	explorer.arc.nasa.gov (128.102.32.18)
Directorio:	Varios.
Contenido:	Ficheros GIF (imágenes del Mariner, Voyager, sonda Magallanes,Galileo, Pioneer, Shuttle, y misiones).
Dirección:	faui43.informatik.uni-erlangen.de (131.188.31.3)
Directorio:	/mounts/epix/public/pub/amiga/aminet/gfx/anim
Contenido:	Tutti3d.mpg file (señoritas afortunadas de TV Alemana)
Dirección:	hipp.etsu.edu (192.43.199.82)
Directorio:	/pub/photo
Contenido:	3D FAQ, imagenes escaneadas en 3D, software de generacion SIRD.
Dirección:	katz.anu.edu.au (150.203.7.91)
Directorio:	/pub/stereograms/images/*
Contenido:	SIRDS y archivos/programas asociados.
Dirección:	moink.nmsu.edu (128.123.4.58)
Directorio:	/rec.photo
Contenido:	Ficheros tipo FAQ sobre fotografía, filtros, ecuaciones DOF, filtros, información de Nikon y Canon, 3D.
Dirección:	morgan.ucs.mun.ca (134.153.2.99)
Directorio:	/pub/stereo, /pub/juggling
Contenido:	Archivos GIF.
Dirección:	pit-manager.mit.edu (18.172.1.27)
Directorio:	/pub/usenet/rec.photo
Contenido:	Una copia del anterior moink.nmsu.edu, llamado rec.photo FAQ, y info sobre formatos medios.
Dirección:	phoenix.oulu.fi (130.231.240.17)
Directorio:	/pub/incoming/EsaKuru3D
Contenido:	180 SIRD JPEG fotos en 3D mapa terráqueo en color.
Dirección:	seds.lpl.arizona.edu (??.??.??.??)
Directorio:	/pub/spacecraft/CLEMENTINE/images
Contenido:	Imágenes del Clementine spacecraft.
Dirección:	sunsite.unc.edu (152.2.22.81)
Directorio:	/pub/academic/computer-science/virtual-reality/3D /pub/academic/stereo-pairs/-{photo,fractals,earth}

→

rán entrar solo en unos momentos dados y por un tiempo definido, ya que somos "invitados".

EJEMPLO DE FTPMAIL

Elegimos un ordenador con FTPmail como por ejemplo el decwrl.dec.com de Digital Equipment Corp, y solicitamos ayuda del correspondiente servidor con el comando help.

Si nuestro identificador es jose@ordena.mired.es, el mensaje que mandaríamos por E-mail seria el Siguiente :

```
From: jose@ordena.mired.es
To: ftpmail@decwrl.dec.com
Subjet: Request for help
help
```

Transcurrido muy poco tiempo, recibimos el siguiente mensaje en nuestro buzón del E-mail en inglés, que resumido es el siguiente:

```
From nobody@ftp-gw-1.pa.dec.com (Cabecera del servidor)
To: jose@ordena.Mired.ES (A quien va dirigido)
```

Y a continuación el resultado de nuestra petición, en este caso el del comando help, produce este texto:

```
— Help —
>>> Los comandos son:
    reply <MAILADDR>
connect [HOST [USER [PASS [ACCT]]]]
    password por defecto : anonymous
    ascii
    binary
    ...
```

Esta ayuda indica entre otras cosas, por ejemplo este help que con el comando "chunksize" el tamaño máximo en bytes para transferencias. Y que, como máximo podemos hacer diez get. Luego vienen unas notas que nos indican posibles errores, si superamos el tamaño o hacemos uso incorrecto de los comandos. Incluso nos indica ejemplos, y que el autor del ftpmail fue Paul Vixie que trabajó en DEC desde 1989 a 1993.

Y acaba dando otro fin de cabecera como este :

```
From: jose@ordena.Mired.ES    (Jose Brown)
To: ftpmail@decwrl.dec.com
Subject: Reques for help
  — End of Request Mail Header —
```

/pub/academic/stereograms
/pub/academic/red_blue
Contenido: Una gran seleccion de archivos GIF estéreos.

Esta lista contiene información sobre hosts con ficheros, preguntas-respuestas tipo FAQ, programas sobre fotografía e imagenes en 3D, accesibles por la internet vía ftp anónimo. La lista está por orden alfabético, si desea enviar alguna corrección, o añadir alguna otra dirección interesante, puede hacerlo a la dirección:

Joel.Alpers@FtCollinsCO.NCR.COM

Estos son algunos de los servidores de FTPmail que existen:

ftpmail@decwrl.dec.com
bitftp@pucc.princeton.edu
bitftp@vm.gmd.de
ftpmail@ftp.uni-stuttgart.de
ftpmail@grasp.insa-lyon.fr
bitftp@plearn.edu.pl
ftpmail@doc.ic.ak.uk
ftp-support@nic.switch.ch

OTROS MUY INTERESANTES

En el servidor de la NCSA (National Center for Supercomputing Applications), llamado ftp.ncsa.uiuc.edu y en el directorio: pc/mosaic/windows podemos encontrar la última versión del programa NCSA Mosaic para Windows, que nos permite trabajar directamente con un programa cliente WWW para acceder a servidores WWW, viendo ficheros y recursos de Internet de manera gráfica.

En el servidor de Microsoft denominado ftp.microsoft.com, en el directorio /softlib/mslfiles hay una versión del anterior programa Mosaic en el fichero PW1118.EXE.

En archive.umich.edu de la Universidad de Michigan hay ficheros relativos a MSDOS y Mac, asi en el directorio /msdos/uploads podemos encontrar muchas utilidades de DOS y de WIN, por ejemplo el Chief's Windows Installer.

En ftp.uu.net, en el directorio /tmp y subdirectorios encontraremos utilidades freeware y shareware muy interesantes.

En ftp.cis.ksu.edu y en el directorio /pub/upload y subdirectorios adjuntos hay programas y utilidades para Windows y DOS.

En ftp.demon.co.uk y dentro del directorio /pub/ibm-pc/windows/utilities hay muchas utilidades para Windows y comunicaciones como un programa de tipo dialer llamado DIALEXE.ZIP .

En micros.hensa.-ac.uk como veremos en este artículo, y dentro del /micros/ibmpc y sus subdirectorios, como /win, hay utilidades y programas de todo tipo incluido el Chief's Windows Installer.

En Finlandia tenemos uno poco conocido hasta ahora que es el garbo.uwasa.fi de la Red Finlandesa de Ciencia con varios directorios muy interesantes.

Algunas empresas como Microsoft ponen también a disposición de los usuarios, servidores de ftp, en este caso, ftp.microsoft.com que tiene en el directorio /peropsys/wfw/tcip para expertos, sysops de BBS y sistemas remotos bastante información y versiones beta de programas, como el fichero MTCB3.EXE en el anterior directorio de ../tcip/vxdbeta.

Para acceder a Internet hay que contratar el servicio a través de empresas como Goya Servicios Telemáticos o Telefónica

También Borland ofrece un servidor de ftp en ftp.borland.com.

Sin olvidar, por supuesto, el clásico de la Finland Research Network llamado ftp.funet.fi (128.214.1.1), que cuenta con cerca de 10 Gb de ficheros diversos de MSDOS, WIN, UNIX y programas de dominio público para PC,Mac,Amiga,Atari, etc.

Podemos encontrar las normas de la organización CERT sobre Internet, seguridad informática con su servidor cert.org. Si lo que desea es código fuente y news Usenet, puede acceder a ftp.uu.net.

Algunos servidores con ftp anonymous se especializan más en unos temas que en otros, así podemos citar los siguientes:

```
MENU DE CORREO                                    EUnet
1. Leer mensajes pendientes
   (Para leer los mensajes que han lle-
   gado nuevos desde la última vez)
2. Leer mensajes archivados
   (Para leer los mensajes recibidos
   que Vd. archivó)
3. Agenda
   (Para editar su agenda personal)
4. Editar mail
   (Para enviar mensajes)
5. Ayuda
   (Describe las opciones de este menú)
0. Menu anterior
```

Figura 1.

rascal.ics.utexas.edu en el directorio /misc/av hay programas y ficheros con textos de bromas,humor en cabina de avión en vuelo.

En el servidor ftp.iup.edu en /fligtsim/edu veremos que simuladores de vuelo y escenarios de aviación.

rftm.mit.edu y a través del subdirectorio /pub/usenet/rec.autos/* y /usenet/news.answers podemos ver listas de correo, archivos, guías de usuario, leyes de California, listas de fabricantes de motores y otros, relacionados con el automóvil.

onion.rain.com en /pub/falcon3/* podemos encontrar teoría de simulación de vuelo, escenarios de simuladores de vuelo.

Si es aficionado al Cyberpunk puede consultar etext.archive.umich.edu, directorio /pub/zines/future.culture/ hay una serie de preguntas y respuestas FAQ sobre el tema. También en rftm.mit.edu, en el directorio /pub/usenet/news.answers/music/ hay documentos que tratan de explicar los mismos temas musicales, incluso industriales del Cyberpunk.

FTP EN ESPAÑA

Aunque no hay demasiados, su número se incrementa día a día, podemos citar los siguientes:

asterix.fi.upm.es	(138.100.8.6)
goya.eunet.es	(193.127.1.2)
ftp.rediris.es	(130.206.1.1)
sqdxo1.sq.ehu.es	(158.227.108.30)
diable.upc.es	(147.83.36.47)

Mención aparte podemos hacer, de asterix.fi.upm.es (que algunos lectores ya conocerán, en /pub/docs disponemos de documentos con las reglas de Internet.

También algunas BBS nos dejan acceder a Ftpmail, como las integrantes de INTERRED (mazanet.es), a través de jjamor@gedeon.ls.fi.upm.es.

En el fichero ESPAÑA.TXT pueden ver organizaciones y direcciones Internet españolas, incluso algún postmaster del dominio en concreto, sacado de ficheros públicos de la empresa Goya Servicios Telemáticos S.A, incluso servidores gopher, finger y Bibliotecas Españolas procedente de información de la EUITI de Zaragoza.

OBTENER EL CONTENIDO DE UN DIRECTORIO

A través del correo vimos como el fichero fliexe10.zip se encontraba, entre otros muchos en micros.-hensa.ac.uk, en el subdirectorio /mirrors/simtel/msdos/animate. Veamos primeramente como obtener el contenido de este subdirectorio, en el que naturalmente debe encontrarse el fichero mencionado. Para ello deberemos poner el siguiente mensaje:

```
From: josé@ordena.mired.es
To: ftpmail@decwrl.dec.com
Subjet:
reply josé@ordena.mired.es
connect micros.hensa.ac.uk
ascii
chdir /mirror/simtel/msdos/animate
dir
quit
```

Enviado el mensaje y transcurridos unos minutos recibimos otro, en el que, entre otras cosas se nos confirmaba la llegada de nuestro mensaje, y el número que teníamos en la lista de espera. (Ver listado 1).

Antes de traernos el fichero objeto de la consulta y, con el fin de que esta explicación resulte más coherente, veamos la manera de traernos el primer fichero de texto (ascii) 00-index.txt, que se encuentra en el subdirectorio. Para ello deberemos mandar el siguiente mensaje:

```
From: josé@ordena.mired.es
To: ftpmail@decwrl.dec.com
Subjet:
reply josé@ordena.mired.es
connect micros.hensa.ac.uk
ascii
chdir /mirror/simtel/msdos/animate
get 00_index.txt
quit
```

Como en el caso anterior, primero recibiremos un mensaje anunciando la llegada y demás detalles, horas mas tarde recibiremos el fichero de texto.

OTRO EJEMPLO REAL

Supongamos ahora que queremos recibir el fichero fliexe10.zip que según se ha visto se encuentra en el subdirectorio /mirror/simtel/msdos/animate de micros.hensa.ac.uk.

En este caso, se trata de un fichero "binario" y, como solamente se pueden enviar por correo ficheros ascii, habrá que incluir en el mensaje la orden "uuencode", que transforma los ficheros binarios a ascii.

Naturalmente, una vez recibido el fichero, habrá que transformarlo nuevamente de ascii a binario mediante el programa "uudecode".

Por consiguiente, el mensaje a enviar será el siguiente:

```
From: josé@ordena.mired.es
To: ftpmail@decwrl.dec.com
Subjet:
reply josé@ordena.mired.es
connect micros.hensa.ac.uk
binary
uuencode
chdir /mirror/simtel/msdos/animate
get fliexe10.zip
quit
```

LISTADO 1

```
From nobody@ftp-gw-1.pa.dec.com
Return-Path: <nobody@ftp-gw-1.pa.dec.com>
Received: from ftp-gw-1.pa.dec.com by ordena.Mired.ES (IBM OS/2 SENDMAIL 1.2.10/1.0)

<<< dir
>>> list directory
<<< quit
X-Service-Address: ftpmail@ftp-gw-1.pa.dec.com
X-Job-Number: 783530169.10992

y unas horas mas tarde recibimos la respuesta

Precedence: bulk
Reply-To: <nobody@ftp-gw-1.pa.dec.com>

DIR

-r—r—r—     1 syspds   syspds   1654 Oct  5  1994 00_index.txt
-rw-r——-    1 syspds   syspds   80520 Mar 30  1992 fgl110c.zip
-rw-r——-    1 syspds   syspds   224716 Mar 31  1994 flicit24.zip
-rw-r——-    1 syspds   syspds   225947 Apr  7  1994 flicit25.zip
-r—r—r—     1 syspds   syspds   39899 Apr 29  1994 -fliexe10.zip- <—
-rw-r——-    1 syspds   syspds   50789 Apr 28  1991 flimaker.zip
-rw-r——-    1 syspds   syspds   165586 Jul 16  1989 gmovie.zip
-rw-r——-    1 syspds   syspds   152660 Jun 30  1993 hcmake11.zip
-rw-r——-    1 syspds   syspds   37115 Jun 30  1993 hcplay12.zip
-rw-r——-    1 syspds   syspds   88081 Oct 16  1993 nbird20.zip
-rw-r——-    1 syspds   syspds   209880 Mar 18  1991 pdgrasp.zip
-rw-r——-    1 syspds   syspds   4610 May 10  1990 uc3d.zip
-rw-r——-    1 syspds   syspds   91441 Dec  4  1993 winter11.zip
```




```
CONTENIDO DEL MENSAJE                          EUnet
Date:  Tue,  13 Dec 94 03:49:49 -0800
From:  "ftpmail service on ftp-gw-1.pa.de
c.com" <nobody@pa.dec.com>
To:  pacov@intertex.es
Subject:  your ftpmail request has been r
eceived [HELP]
   -- Help --
>>> ¤Id: help-text,v 1.7 1993/05/05 00:4
9:43 vixie Exp ¤
>>>
>>> ftpmail is not a supported service.
From time to time it stops working;
>>> we will tend to it when we get the t
ime. Outages of a week or more are not
#→PAG.SIGUIENTE *#→PAG.ANTERIOR  0→SALIR
1→MENS.SIG         2→MENS.ANT    6→BORRAR
3→IMPRIMIR  4→GUARDAR  5→GUARDAR BINARIO
7→CARGAR 8→RESPONDER 9→LISTA DE MENSAJES
```

Al igual que en el caso anterior, se recibió casi de inmediato un mensaje informándonos de las condiciones del envio y demás pormenores. Como ya sabíamos, el fichero tiene menos de 64K y por tanto el ftpmail fué enviado en un solo mensaje. Como ya se sabe, los mensajes de más de 64K se envian en dos o más partes debiendo luego unirse y transformarse de ascii a binario.

Finalmente hemos querido traernos todo el contenido de todos los subdirectorios de /mirror/simtel/msdos , ya que como es fácil de adivinar se trata del conocido CDROM Simtel. Para ello se envió el siguiente mensaje:

```
From: josé@ordena.mired.es
To: ftpmail@decwrl.dec.com
Subjet:
reply josé@ordena.mired.es
connect micros.hensa.ac.uk
ascii
chdir /mirror/simtel/msdos
dir *
quit
```

Como siempre, recibimos la confirmación del envío, y más tarde el contenido de todos los subdirectorios. Dada la gran extensión de estos subdirectorios el envio se realizó en 14 partes.

Existen otros servicios que permiten, por ejemplo, conversar con cientos de personas simultáneamente

Como puede usted ver, aparece siempre un directorio denominado mirror, es "espejo" de programas de otros servidores. Normalmente hay directorios llamados mirror u otros públicos indicados como /pub/ y algunas veces se indica /pc/ u otra indicación como /windows/ que nos da más pistas. En próximos artículos intentaremos dar más métodos y direcciones, incluso más guías parecidas a las telefónicas para no ir tan a ciegas.

Sólo Programadores agradece a la EUITIZ de Zaragoza , cátedra de REDES y en especial a Antonio Montañés, la colaboración prestada para la elaboración de este artículo. ■

FIGURA 2

COMANDOS DE FTPMAIL

Las comandos más fundamentales que nos servirán para hacer la conexión, visión, transferencia y trabajo en sesión Ftpmail se muestran a continuación:

Especificar la dirección de correo

reply "dirección" Dirección de correo para respuesta

Conectar con un host.-

connect "host" Conectar con la dirección especificada

Configurar opciones.-

ascii Para archivos de texto
binary Para archivos binarios
uuencode Convierte archivos binarios a texto

Especificar directorios.-

chdir "directorio" Cambiar al directorio especificado

Solicitar archivos.-

get "archivo" Enviar el archivo especificado

Solicitar información.-

help Enviar descripción de uso del Ftpmail
dir "directorio" Enviar un directorio completo
ls "directorio" Enviar un directorio breve

Terminar la sesión.-

quit Terminar la sesión

Con el programa FTP se puede acceder de forma remota a ordenadores muy distantes

Direcciones de interés

EUnet
Goya Servicios Telemáticos S.A.
C/Clara del Rey, nº 8, 1º-7
Tel: (91) 413 48 56
Fax: (91) 413 49 01
e-mail: jae@Spain.EU.net

DETECCION DE LA CPU EN EL PC

David Brioso

Uno de los problemas más comunes que se plantean, al iniciar la creación de cualquier programa, es la elección del microprocesador en el que éste va a ejecutarse. Esta es una de las tareas rutinarias que se realizan al comprobar la configuración del ordenador, junto con la cantidad de memoria disponible, la tarjeta de vídeo, ratón, etc.

La mayoría de los compiladores modernos ofrece la posibilidad de generar código ejecutable para cualquiera de los microprocesadores de la familia Intel a partir de un listado fuente común. Es por esto, y por el abanico cada vez mayor de modelos disponibles, por lo que resulta de vital importancia detectar la presencia de la CPU antes de ejecutar el programa en cuestión.

Para superar la falta de documentación sobre la materia y aclararla de una forma práctica vamos a estudiar a fondo los tres métodos que aparecen con más frecuencia en el proceso de detección del microprocesador del PC. La forma más sencilla de comprender el mecanismo de funcionamiento de estos tres algoritmos es analizarlos en un estado casi puro, es decir, en Ensamblador, que podrán ser adaptados a cualquier lenguaje que tenga una interfaz de bajo nivel, como explicaremos en la última parte del artículo.

EL PUNTERO DE PILA

El primero de estos algoritmos, también el más simple, distingue una CPU 8088/86 de las superiores. De esta manera podemos averiguar si el ordenador dispone de un chip 286 o posterior, para aquellos programas que precisen uno de estos procesadores para funcionar. Este método se basa en una pequeña diferencia existente en la forma en que la CPU trata el registro puntero de pila (SP, Stack Pointer). Cuando los modelos más antiguos de Intel ejecutan la instrucción PUSH SP, incrementan el puntero y, a continuación, almacenan el nuevo valor. Las versiones posteriores invierten el orden de operación, almacenando en primer lugar el puntero de la pila, e incrementándolo posteriormente.

EL REGISTRO DE FLAGS

El siguiente sistema de detección es capaz de distinguir los tres modelos más antiguos de la serie 80x86 de Intel, el 8086, 80286 y 80386. Evidentemente, cualquier procesador superior a éste último será tomado como un 386, por aquello de la compatibilidad descendente. El registro de flags de la CPU conserva los bits indicativos de las condiciones: cero, acarreo, overflow, etc. Este registro ha sido extendido a medida que crecían las necesidades de los nuevos microprocesadores. Si comprobamos la disponibilidad de algunos bits de dicho registro podremos averiguar el modelo de CPU del que se trata.

Este método quizá sea el más complejo de los tres, por lo que nos extenderemos más en su análisis. En el segundo listado aparecen una serie de manipulaciones sobre el registro de flags que nos informarán del modelo de procesador con una cierta precisión. Este algoritmo fuerza al registro a adoptar un valor que será rechazado si la versión del microprocesador no corresponde con la deseada. Si éste no soporta los bits añadidos al registro por las versiones superiores, éstos permanecerán activados (a uno) aunque intentemos modificar su valor directamente.

El camino más corto para alterar el contenido del registro de flags de la máquina es a través de la pila, que nos permite cargar en él cualquier valor. Nuestro algoritmo aprovecha esta posibilidad para modificar los bits que permanecen sin usar en un 8086 y, si no es el caso, también los del 80286. Esta forma de detectar la CPU del ordenador adolece de un defecto que, según las necesidades de cada programa, puede resultar determinante o no para su uso:

no tiene en cuenta los modelos más recientes de la serie de Intel. Por lo tanto, si es estrictamente necesario obtener una información más exacta, el último algoritmo es el adecuado.

LAS EXCEPCIONES DE LA CPU

Las excepciones del microprocesador son, seguramente, la forma más fiable de interrogar al propio chip para que nos entregue la información que buscamos. Estas excepciones, que no son otra cosa que una interrupción de tipo hardware, se producen cada vez que la CPU encuentra un código de instrucción que no puede decodificar y ejecutar. Esto es lo que ocurre siempre que se ejecuta un programa creado para 386, en un PC/XT, que sólo dispone de un modesto 8086. En este caso, el microprocesador desconoce las instrucciones que fueron añadidas con posterioridad a sus hermanos mayores. Al no saber qué hacer con ellas, genera una señal de interrupción 06h para informar del error que se ha producido. La rutina de servicio de la interrupción puede emular por software la instrucción, una técnica que se emplea en el desarrollo de nuevos chips, imprimir un mensaje de error o, simplemente, descartarla y ejecutar la siguiente. Esta última posibilidad no es muy recomendable porque puede producir resultados inesperados, ya que la instrucción supuestamente ejecutada debería haber modificado de alguna manera el curso del programa.

Nuestro algoritmo captura el vector de la interrupción 06h y lo desvía hacia la rutina que se encarga de comprobar el tipo de CPU. Después se ejecuta una instrucción de un modelo en concreto y, dependiendo del resultado, sabremos si se trata de él o no. Si la instrucción es ejecutada con toda normalidad, se puede tratar del modelo que buscamos o de uno superior. Si se genera la excepción, se trata de un chip inferior. De igual manera, si extendiéramos el proceso a todos los modelos, podríamos averiguar con cuál de ellos estamos tratando.

Algunos ensambladores no permiten la generación de código objeto para los últimos miembros de la familia Intel. En estos casos, el único medio para poder utilizar este algoritmo es mediante el ensamblado manual de las instrucciones de prueba.

La principal ventaja de este último método es que puede ser ampliado a las nuevas CPUs de la familia que puedan aparecer en un futuro más o me-

El registro de FLAGS es capaz de distinguir los tres modelos más antiguos de la serie 80x86 de Intel

CUADRO 2

```
cpu2    PROC
pushf                       ; Copia el registro de flags en
pop     bx                  ; el registro BX
and     bh,0Fh              ; Borra los 4 bits más altos de BX
push    bx                  ; Vuelve a guardar BX en el stack
popf                        ; y lo lee en el registro de flags
pushf                       ; Copia de nuevo el registro de los
pop     ax                  ; flags, esta vez en AX
and     ax,0F000h           ; Borra los 12 bits más bajos de AX
cmp     ax,0F000h           ; y comprueba si los otros 4 están
je      c_1                 ; activados. Si lo estan, es un 8086
or      bh,0F0h             ; Activa los 4 bits mas altos de BX
push    bx                  ; Guarda el registro BX en el stack
popf                        ; y lo recoge en el registro de flags
pushf                       ; Vuelve a guardar el registro de flags
pop     ax                  ; y lo recupera en el registro AX
and     ax,0F000h           ; Borra los 12 bits más bajos de AX
jz      c_2                 ; y, si los otros 4 estan encendidos,
                            ; es un microprocesador 80286
c_3:    mov         ax,3         ; Si no es ni uno ni otro,
                                 se trata de
ret                         ; un 80386, 486 o superior
c_2:    mov         ax,2         ; Devuelve en AX el
                                 codigo de CPU 386
ret                         ;
c_1:    mov         ax,1         ; Devuelve en AX el
                                 codigo de CPU 8086
ret                         ;
cpu2    ENDP
```

Listado perteneciente al segundo algoritmo de detección del microprocesador del PC. El registro AX devuelve 1 si es un 8086, un 2 si es un 286, y un 3 si es un 386 o superior.

CUADRO 1

```
cpu1  PROC
mov             ax,sp           ; Copia el valor actual de SP en AX
push            sp              ; Guarda el registro puntero en el stack
pop             bx                          ; Recoge el valor en BX, para compararlo
cmp             ax,bx           ; con el valor anterior
je              cp_2            ; Si coinciden, la CPU es un 286 o
cp_1:mov        ax,1            ; superior
ret                                         ; Regresa con el código de CPU en AX
cp_2:mov        ax,2            ; Si no coinciden, la CPU es un 8086
ret                                         ; Regresa con el código de CPU en AX
cpu1  ENDP
```

Listado perteneciente al primer algoritmo de detección del microprocesador del PC. El registro AX devuelve 1 si es un 8086 y un 2 si es un 286 o superior.

CUADRO 3

```
cpu3    PROC
xor     ax,ax             ; Carga ES con el segmento de la
mov     es,ax             ; tabla de vectores de interrupciones
mov     bx,0018h          ; BX apunta al vector de int 06h
mov     ax,es:[bx]        ; Lee el offset de la interrupción y
push    ax                lo guarda en el stack
mov     ax,es:[bx+2]      ; Lee el segmento de la interrupción
push    ax                y lo guarda también en el stack
mov     ax,offset int6    ; Ahora escribe la dirección de la
mov     es:[bx],ax        ; rutina de interrupción en el vector
mov     ax,cs             ; para que salte allí si se produce
mov     es:[bx+2],ax      ; una excepción en el microprocesador
NOP                       ; Aquí se pone la instrucción de
                          prueba
pop     ax                ; Recupera del stack el segmento del
mov     es:[bx+2],ax      ; vector de interrupción original
pop     ax                ; y lo escribe en la tabla, así como
mov     es:[bx],ax        ; el offset
mov     ax,1              ; Devuelve el código de procesador
                          X86
ret                       ;
int6:                     ; Rutina de gestión de la excepción:
pop     ax                ; Recupera del stack las tres palabras
pop     ax                ; almacenadas al generar la excepción
pop     ax                ; del error
pop     ax                ; Recupera del stack el segmento del
mov     es:[bx+2],ax      ; vector de interrupción original
pop     ax                ;y lo escribe en la tabla, así como
mov     es:[bx],ax        ; el offset
mov     ax,0              ; Devuelve el código de procesador
ret                       ; equivocado
cpu3    ENDP
```

Tercer algoritmo de detección del microprocesador del C.

CUADRO 4

```
#include <stdio.h>

void    main()
{
int cpu;
asm     {
        pushf
        pop     bx
        and     bx,0FFFh
        push    bx
        popf
        pushf
        pop     ax
        mov     cx,8086
        and     ax,0F000h
        cmp     ax,0F000h
        je      stcpu
        mov     cx,286
        or      bx,0F000h
        push    bx
        popf
        pushf
        pop     ax
        and     ax,0F000h
        jz      stcpu
        mov     cx,386
        jmp     stcpu
        }
stcpu: asm      mov     cpu,cx
printf("La CPU del ordenador es: %d\n",cpu);
}
```

Segundo algoritmo de detección del microprocesador del PC.

nos cercano. Los microprocesadores que funcionan por emulación no devolverán una información fiable. Un ejemplo bastante claro es el chip PowerPC, que emula por software al procesador de destino. El único inconveniente de este último sistema de detección, aunque mínimo, es la necesidad de "ensuciarse las manos" con interrupciones y similares, que pueden resultar un poco incómodas de manejar en algunos lenguajes de alto nivel.

LA ADAPTACIÓN A OTROS LENGUAJES

La implementación de los algoritmos en otros lenguajes no tiene por qué suponer una dificultad. Cualquier lenguaje que acepte instrucciones de ensamblador en línea, o que pueda ser enlazado con objetos externos puede resultar adecuado. Además, los listados en ensamblador no requieren apenas modificaciones a la hora de ser añadidos a otro programa, salvo que sean ensamblados aparte. Para ilustrar mejor la adaptación de los algoritmos de detección, hemos incluido el listado en C de uno de ellos.

Los ensambladores en línea pertenecientes a algunos compiladores no permiten la utilización de etiquetas locales dentro del listado fuente en Ensamblador, así que es necesario recurrir a etiquetas C situadas fuera del mismo, esas reliquias que fueron creadas para la función goto y que han quedado prácticamente desprovistas de toda utilidad.

CONCLUSIÓN

Para ampliar la documentación sobre este tema se puede recurrir a Internet o a BBS dedicados a la programación y a la información técnica existentes en nuestro país. Otra fuente de datos son las propias hojas de los fabricantes de microprocesadores compatibles con la familia Intel 80x86. ■

LISTAS Y VENTANAS, UNA MIRADA MÁS A FONDO

Bernardo García



El aspecto gráfico no lo es todo en el entorno Windows. Debajo de su apariencia externa se esconden otras muchas posibilidades para el programador.

En el artículo del mes pasado se comenzó la creación de una macrolista de tareas. Esta aplicación va a permitir conocer con más profundidad la relación entre las distintas aplicaciones que se encuentran en Windows. Aunque generalmente se hace referencia a un solo programa, este entorno permite que varias aplicaciones o tareas se encuentren funcionando, relativamente, al mismo tiempo.

MULTITAREA COOPERATIVA

Esta capacidad se conoce como multitarea, frente al funcionamiento tradicional de las aplicaciones en el entorno PC, conocido como monotarea. Ambos términos se refieren a la cantidad de programas que están siendo ejecutados simultáneamente en un ordenador. Sin entrar en disquisiciones filosóficas, se considera que dos aplicaciones se encuentran funcionando al mismo tiempo cuando ambas se turnan en la utilización de la CPU en intervalos de tiempo muy pequeños. Este tipo de funcionamiento es el que percibe el usuario, ya que al disponer sólo de un procesador, en un instante específico de tiempo sólo puede estar dedicado a una tarea, pero alternando cada una de ellas da la impresión, de que están funcionando de forma simultánea.

Dentro de lo que puede considerarse como multitarea se suelen diferenciar varios tipos. En primer lugar la real, que sólo es posible conseguir cuando se utilizan varios procesadores. La multitarea simulada puede ser preemptiva, del termino inglés preemptive, en cuyo caso el tiempo es repartido de forma mas o menos equitativa entre los diferentes programas, por otro proceso independiente. Otro de los métodos empleados es la multitarea cooperativa, en la que los procesos colaboran unos con otros cediéndose el tiempo.

El sistema operativo DOS es monotarea, ya que sólo se puede ejecutar un programa cada vez. Windows emplea los dos tipos de multitarea antes comentados. Internamente utiliza método preemptivo, mientras que entre las aplicaciones se emplea el cooperativo. Es decir los programas que se encuentran funcionando en este entorno cooperan entre si para funcionar todos a la vez.

Durante el tiempo que una aplicación esta ejecutándose, puede estar compartiendo los recursos del ordenador con otras. Visualmente cada tarea se identifica con una ventana, aunque un mismo programa puede crear mas de una ventana o ninguna.

LA LISTA DE TAREAS

El propio entorno permite al usuario cambiar entre una y otra aplicación utilizando la Lista de Tareas, accesible pulsando simultáneamente las teclas de Control y Escape.

A diferencia de la versión estándar, que sólo muestra los títulos de las ventanas principales de las aplicaciones, es decir, aquellas cuya ventana padre es el propio Windows, la macrolista va a permitir observar otros elementos, como botones, editores, etc. que también son, aunque no lo aparenten, ventanas.

Para empezar se han realizado ciertos cambios en el aspecto del programa, como puede verse en la figura 1, empleando un editor de recursos, que como se ha visto es el método mas sencillo para el diseño de las ventanas de una aplicación.

A grandes rasgos la presentación no ha cambiado. Se siguen teniendo dos botones radiales para mostrar una lista o un combo con las ventanas indica-

das. Los botones de Ok y Cancel, han cambiado sus nombres por Activar y Salir, aunque conservan su funcionalidad. La lista y el combo siguen estando ahí, aunque por motivos estéticos ocupan ahora la misma posición en el dialogo, como se ve en la figura 2.

Se han añadido una serie de botones y textos, cuyo funcionamiento se comentara mas adelante, ya que constituye el objetivo de la aplicación.

IDENTIFICANDO LAS VENTANAS

El primer paso consiste en rellenar la lista con todas las ventanas que se encuentren. Para ello se realizo en el mes anterior una función, RellenarLista(), que busca las ventanas y las añade a una lista. Y su hermana RellenarCombo(), que realiza la misma operación sobre un combobox, identificando sólo las ventanas principales de las aplicaciones. El método empleado en cada función es diferente, veamos con mayor detalle como realizan su tarea cada una.

Para recorrer la lista de ventanas, RellenarLista(), comienza llamando a GetWindow(), que devuelve un handle a la ventana que cumpla con las características indicadas. Esta función recibe dos valores, en primer lugar el identificador de la ventana inicial y un valor que determina la relación entre ambas, en este caso

```
CurrWnd = GetWindow(hWnd, GW_HWNDFIRST);
```

devuelve el valor de la primera ventana principal de la lista. Si la llamada a la función fallara, el valor que retorna es NULL, permitiendo así identificar cuando no se ha encontrado la ventana buscada.

El objetivo perseguido al guardar la información de las ventanas es poder utilizarla posteriormente para localizar estas sin volver a recorrer toda la lista. Una vez que se ha obtenido el handle de una ventana, se necesita el titulo de la misma y la clase a la que pertenece para poder buscarla directamente con la función FindWindow(). Por esta razón se guardan estos valores en la lista, obteniendo el título con GetWindowText() y la clase con GetClassName(). Ambas funciones necesitan el mismo número de parámetros, para empezar el identificador de la ventana de la que se quiere obtener la información, en este caso CurrWnd. Los demás valores indican donde se deben guardar los valores obtenidos, en las variables nombre o clase, y la longitud máxima que se puede guardar en estas variables. Con los dos valores obtenidos por este método, se construirá el texto que mas adelante servirá para identificar a la ventana.

Al estar construyendo una lista con todas las ventanas existentes, se debe tener en cuenta la relación de parentesco entre estas. De esta forma, cada ventana puede ser "padre" de más ventanas, por lo que se realiza otra comprobación con cada handle, llamando a

```
ChildWnd = GetWindow(CurrWnd, GW_CHILD);
```

para averiguar si existen "hijas" de la ventana que se esta tratando, y en caso afirmativo añadirlas a la lista junto con las demás. Se debe observar que en primer caso se utiliza hWnd y en el segundo CurrWnd para la ventana inicial de la búsqueda, haciendo que ambas sean diferentes al comenzar a buscar en posiciones diferentes de la lista de ventanas.

En ambos casos, para continuar recorriendo la lista es necesario encontrar el siguiente elemento, para lo que solo es necesario volver a llamar a GetWindow(), pero en esta ocasión utilizando la constante GW_HWNDNEXT como valor para relacionar las ventanas. Como puede suponerse existen otros valores posibles para las búsquedas, que pueden verse en la figura 3.

Un caso especial lo constituyen las ventanas que carecen de título. Aunque aparecen al recorrer la lista con GetWindow(), en el caso de las ventanas principales Windows no las incluyen en su lista de tareas, principalmente por la dificultad que entraña su búsqueda.

Aunque en esencia funciona igual que la versión del articulo anterior, se han realizado algunos cambios sobre la función RellenarLista(). Para empezar se incluyen las ventanas principales sin titulo, aunque no es habitual que existan, realizando además la búsqueda de sus ventanas hijas.

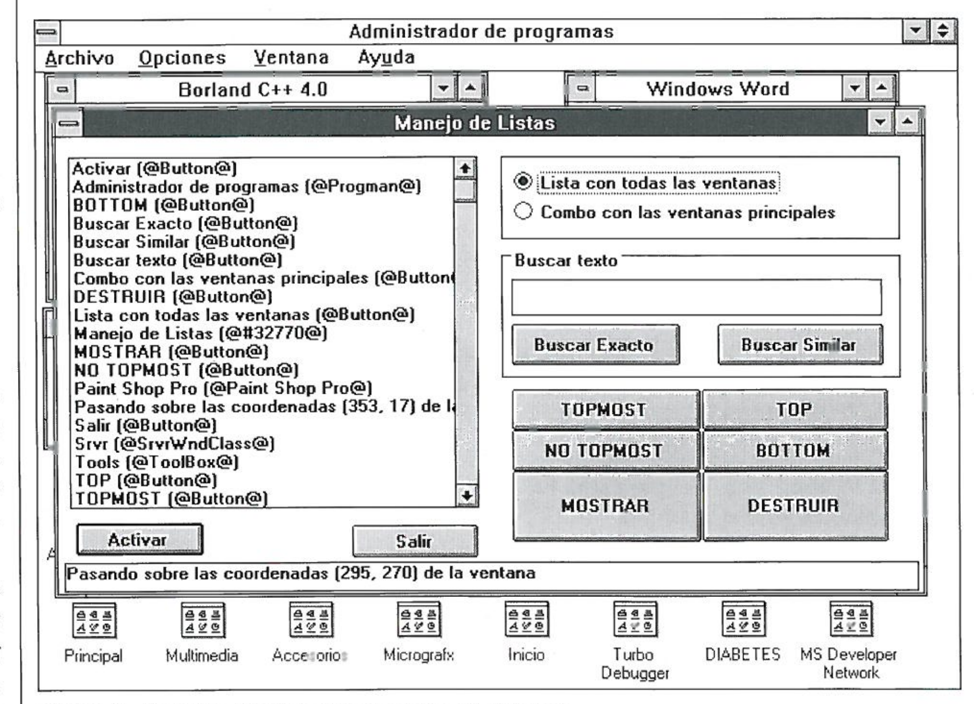


Figura 1.- Aspecto externo del programa de ejemplo.

Para permitir la búsqueda de cualquier tipo de ventana, se han suprimido los primeros caracteres que se añadían al texto para diferenciar padres e hijas. Este cambio va a permitir manipular cualquiera de las ventanas, ya

que anteriormente solo era posible actuar sobre las principales. Al desaparecer estos caracteres que precedían al nombre, se habilita la posibilidad de localizarlas y probar como reaccionan a las operaciones que se van a explicar.

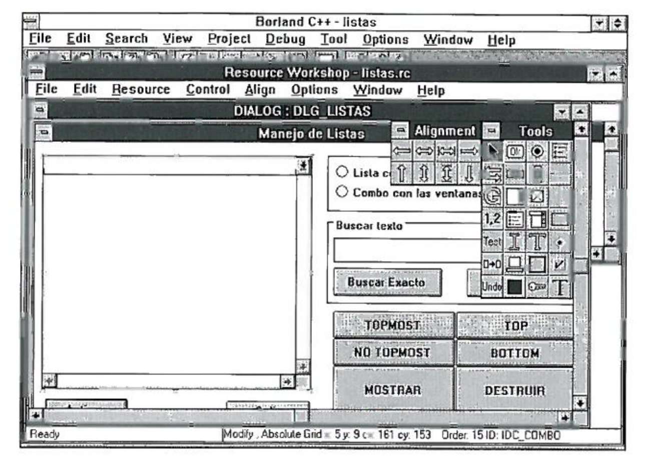
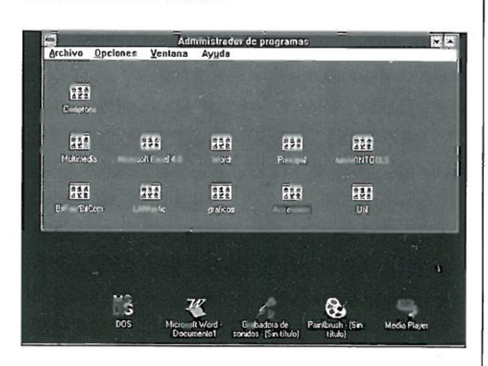


Figura 2.- Lista y Combo ocupan la misma posición en el dialogo.

El método empleado por RellenarCombo() realiza un proceso muy similar al de la lista de tareas estándar de Windows, llamando a la función EnumWindows(). Esta realiza automáticamente la tarea de identificar la ventanas principales de cada aplicación, llamando a una rutina del usuario para cada una de ellas.



En este caso la pantalla no muestra la misma cantidad de información, EnumWindows() solo se ocupa de las ventanas principales. Su funcionamiento es sencillo, el usuario debe facilitar un rutina de tipo callback, refiriéndose a la manera de devolver los valores. Esta función, que en este ejemplo se ha llamado EnumWindowsProc(), es la encargada de manipular los datos de cada ventana que le son facilitados por EnumWindows().

Si se observa detalladamente el funcionamiento de RellenarCombo(), se puede ver como la importancia de esta función se encuentra en la línea

EnumWindows(EnumWindowsProc, hWndCombo);

siendo una función muy reducida en relación a RellenarLista(). La otra parte del proceso, la encargada de construir el texto que se debe añadir al combobox, esta en EnumWindowsProc(). El algoritmo interno es idéntico al empleado en la lista, exceptuando el hecho de que la búsqueda de la ventana es realizada antes de entrar en la función, de forma que internamente no se debe preocupar de si hay o no mas ventanas.

BUSCANDO LAS VENTANAS

Con la lista o el combobox ya rellenados, solo resta implementar las diversas operaciones que se desea realizar con las ventanas. Todas estas poseen un elemento común, el primer paso es encontrar el handle de la ventana.

Esta operación se realiza en la función ObtenerHandle(). El procedimiento consiste en buscar la ventana utilizando la función FindWindow(), para lo que es necesario conocer el titulo y la clase de la ventana. Ambos valores están guardados en los textos que componen la lista y el combobox, por lo que es necesario extraerlos de su posición.

Se plantea un problema, debido a que se tienen dos elementos diferentes de los que se desea obtener un texto. El manejo de las funciones apropiadas difiere entre ambos. Después de haber determinado cual de los elementos se esta manejando, gracias a Button_GetCheck(), una de las macros incluidas en windowsx.h, se debe averiguar si hay algún elemento seleccionado, proceso idéntico para ambos.

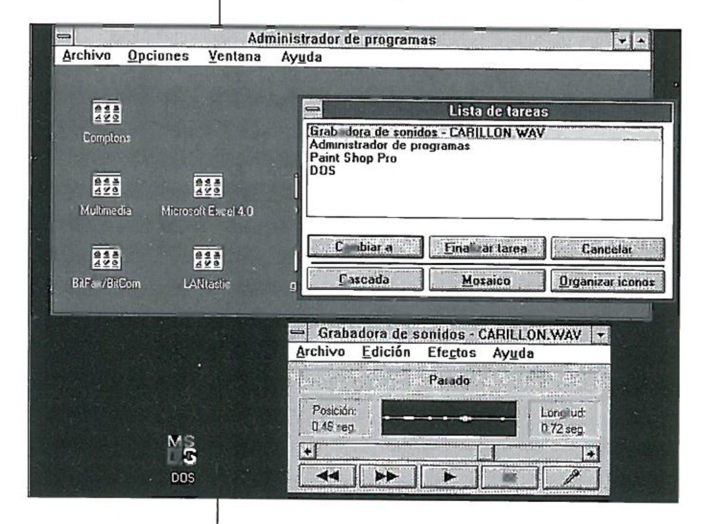
Si se tiene alguna posición seleccionada, tanto ListBox_GetCurSel() como ComboBox_GetCurSel() devuelven un valor diferente de -1.

En el caso de las listas esto indica que hay un elemento seleccionado, cuyo texto puede ser obtenido con ListBox_GetText() para localizar la ventana.

Los combobox por el contrario, pueden estar en tres posibles estados. Si el valor devuelto es diferente de -1, indica la existencia de un elemento seleccionado, pudiendo recuperar su valor con ComboBox_GetLBText().

En otro caso, la lista asociada al combobox no esta marcada, pero el campo de edición puede contener un valor anterior, por lo que se debe contemplar esta segunda alternativa y obtener el texto con GetWindowText().

En cualquiera de los tres casos, una vez recuperado el texto, se em-



plean varias funciones de C para desmenuzarlo y extraer los valores que interesan para localizar la ventana.

BUSCAR TEXTOS

Se han añadido diversas operaciones a las que ya se incluyeron en el artículo anterior. La primera que se va a ver es la búsqueda de textos. En la pantalla se puede ver un campo de edición donde escribir el texto que se

Valor	Significado
GW_CHILD	Primera ventana hija.
GW_HWNDFIRST	Primera ventana principal de la lista.
GW_HWNDLAST	Ultima ventana principal de la lista.
GW_HWNDNEXT	Siguiente ventana en la lista.
GW_HWNDPREV	Anterior ventana en la lista.
GW_OWNER	Ventana propietaria de la actual.

Figura 3.- Posibles valores para buscar con GetWindow().

desea buscar, y dos opciones, búsqueda exacta y similar.

En este caso, como al obtener el handle de la ventana, es necesario conocer el tipo de elemento, lista o combobox, sobre el que se busca, para emplear una u otra función, aunque sus nombre solo difieran en el comienzo.

El cometido de estos dos botones es buscar y si existe seleccionar el elemento que contenga el texto indicado. En el caso de la búsqueda similar se empleará el mensaje LB_SELECTSTRING o CB_SELECTSTRING, convenientemente camuflados como ListBox_SelectString() y ComboBox_SelectString(), que buscan y seleccionan un texto similar al indicado. Para realizar la búsqueda exacta se hace necesario emplear una combinación de mensajes, LB_FINDSTRINGEXACT y LB_SETCURSEL, pero empleando las macros ListBox_FindStringExact() y ListBox_SetCurSel() para dar un aspecto más agradable al listado. Ambos mensajes existen también para los combobox, con la sustitución de las palabras ListBox por ComboBox y LB por CB en los nombres de las funciones y mensajes.

Con estas funciones, escribiendo un texto y pulsando cualquiera de los botones de búsqueda, se puede localizar rápidamente una ventana.

OPERACIONES ESPECIALES

Aunque parezca un grupo de combate, nos estamos refiriendo al resto de los botones del programa. Estos representan una serie de operaciones que podemos realizar con las ventanas.

Comenzaremos por TOPMOST y NO TOPMOST. Ambas opciones indican un estado, mas que una función, de las ventanas. Estas se encuentran apiladas unas encima de otras, como papeles encima de un escritorio. Al indicar que una ventana está por encima de todas las demás, TOPMOST, se está evitando que otras aparezcan delante suyo. Si se elige una ventana y se pulsa el botón, esta permanecerá siempre visible por encima de las demás. La forma de eliminar este efecto consiste en aplicarle el estado contrario, NO TOPMOST, que permite que otras ventanas escriban encima de ella.

Para conseguir el mismo efecto que NO TOPMOST se puede emplear también la opción BOTTOM, aunque suele ser utilizada para ocultar una ventana o colocarla debajo de todas las demás. Hay que tener cuidado con los efectos de esta opción, ya que si la ventana no esta visible el efecto obtenido es el contrario al esperado.

La siguiente operación es colocar la ventana encima de todas las demás, de forma similar a TOPMOST, pero evitando el inconveniente de no poder ocultar posteriormente la ventana. Con TOP la ventana se hace visible, pero puede ser tapada por otras.

Una opción mas radical la constituye DESTRUIR, que como su nombre indica se ocupa de terminar la ejecución de la ventana seleccionada, enviando el mensaje WM_DESTROY, y releyendo la lista o combobox para reflejar la verdadera situación de Windows.

Tanto la opción de MOSTRAR, como la de Activar, son idénticas, y únicamente realizan la tarea de ceder el control a la aplicación escogida.

Casi todas estas opciones emplean la función SetWindowsPos() para alterar la posición de las ventanas. Se puede observar que esta no es únicamente bidimensional, como parece en un principio, sino tridimensional. Con los parámetros adecuados se pueden variar las coordenadas x e y de la ventana, así como su posición por encima o debajo de las demás, y su tamaño.

CONCLUSIÓN

Aunque las posibilidades son mucho mayores, esta es una pequeña muestra de lo que se puede conseguir manipulando el estado de las ventanas, y que se irá viendo en los siguientes artículos de esta serie. ■

EL EDITOR VI

Fernando J. Echevarrieta

Hasta ahora, se ha realizado una presentación de las características principales de UNIX y se han facilitado los conceptos y herramientas necesarios para situarse y "sobrevivir" en el sistema. Sin embargo, si la misión de un sistema operativo fuera únicamente la de permitir sobrevivir en él, sería un vano desperdicio de esfuerzo y recursos.

Si bien es necesario conocer el marco en el que el usuario debe moverse, así como las reglas fundamentales, la razón del sistema reside en servir de plataforma, para hacer algo de forma activa con la máquina sobre la que se apoya. Por ello, aunque la potencia de todo S.O. reside en el interfaz que presenta con el núcleo, y lo que este último es capaz de hacer, el usuario programador necesitará, al menos, editar sus programas. Además de esto, para cualquier usuario será fundamental disponer de herramientas, que le permitan una participación activa , pues no se limitará a proteger sus propios datos y cambiar este, o aquel fichero de nombre o sitio.

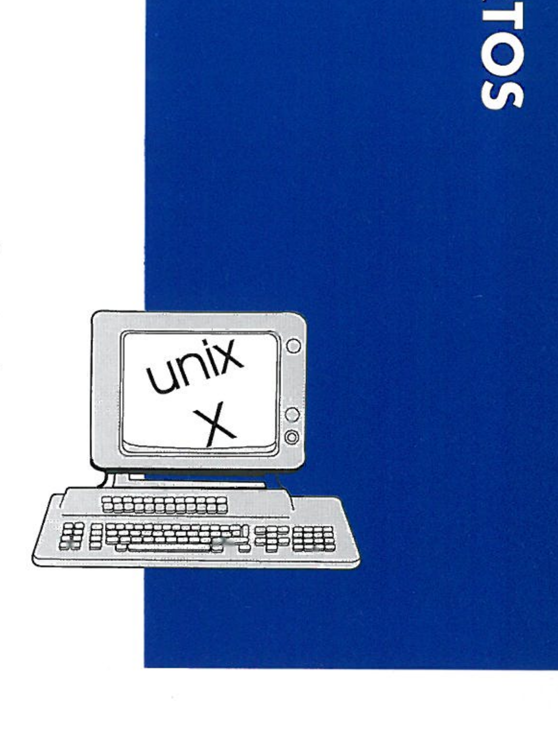
EDITORES DISPONIBLES EN UNIX/LINUX

La principal forma de manejar el entorno es, por supuesto, cambiarlo, primero a nivel de usuario, y más tarde y con más experiencia, realizar completas configuraciones a nivel de administración como root. Para ello será necesario el uso de un editor. Numerosos son los editores que han ido pasando por la historia de UNIX convirtiéndose en estándares: desde los más sencillos, como ed (editor orientado a líneas tan incómodo como el edlin del DOS, aunque mucho más potente), pasando por editores de flujos (streams) como el sed (Stream EDitor), hasta los más complejos, como el gigantesco emacs. El caso del vi, que supuso una revolución en su incorporación a la variante de UNIX de la Universidad de California Berkeley, será el que se tratará más en profundidad en esta serie. Hoy en día, estos dos últimos programas son los estándares más utilizados aunque emacs, por su mayor complejidad, suele presentarse como una parte opcional en un gran número de distribuciones UNIX.

Para comenzar a manejar el entorno será necesaria la utilización de un editor

EL EDITOR VI

El editor vi es un editor orientado a pantalla completa basado en expresiones regulares. Se trata de una herramienta muy potente, muy rápida y, como no, como todo lo potente y rápido en UNIX, muy difícil de aprender. Esta es la razón por la que se han desarrollado algunos editores similares a estándares conocidos del DOS como el joe, muy similar a WordStar, que, como todos los mencionados se encuentra presente en las dos versiones de Linux distribuidas por Sólo Programadores. En vi, la mayoría de los comandos constan de una sola tecla, lo que tiene su origen en las antiguas normas de comunicaciones que como la V23, utilizada hasta el año pa-



En el primer bloque del curso se realizó una toma de contacto con el sistema explicando los principales conceptos para manejarse por él y ofreciendo algunos comandos para ello. A partir de ahora se entra en un nuevo ciclo en el que el usuario pasará a tener una presencia activa.

sado en España para el sistema Ibertex, disponían de un canal rápido en la dirección host-terminal, y de otro muy lento de retorno, por lo que era fundamental que, con unas pocas pulsaciones de tecla se pudieran conseguir grandes efectos. Este es el factor que, a la vez de otorgar tanta potencia

TABLA 1

PRINCIPALES MOVIMIENTOS EN VI

j cursor abajo	k cursor arriba
h cursor izquierda	l cursor derecha
+ línea siguiente	- línea anterior
nl ir a columna n	^ principio de línea
$ última columna	G última línea
b palabra anterior	w palabra siguiente
(frase anterior	) frase siguiente
{ párrafo anterior	} párrafo siguiente
/ búsqueda de	patrón ? búsqueda hacia atrás
n buscar de nuevo	N buscar de nuevo en sentido contrario

al editor, impone tanta dificultad a su aprendizaje.

El editor dispone de dos modos de funcionamiento: a) modo comando y b) modo inserción. Al invocarlo tecleando

vi [lista de ficheros]

aparecerá una pantalla como la que se muestra en la figura 1. Para salir bastará teclear

:q!

desde el modo comando. El programa, al arrancar, lo hará en este modo, en el que se podrá realizar movimientos a lo largo del texto ya escrito, así como introducir comandos precedidos de un número entero, que indica el número de veces que debe repetirse la acción. Para volver a modo comando desde el modo inserción bastará con pulsar ESC. Los principales movimientos son: j (abajo), k (arriba), h (izquierda) y l (derecha). Para manejarlos, una buena práctica es colocar la mano derecha con el dedo índice apoyado sobre la tecla j. Los usuarios que hayan gozado con los juegos de máquinas tan entrañables como el Spectrum o los CPCs, pronto se acostumbrarán a desplazarse por la pantalla con este singular método; el resto, tampoco encontrará demasiados problemas cuando hayan pasado "algunos años". Otros movimientos son los que se encuentran reflejados en la tabla 1. En esta tabla, por patrón se entiende cualquier cadena constante y/o formada por expresiones regulares como las que figuran en la tabla 2.

Para pasar a modo inserción, podrá utilizarse cualquiera de los comandos de inserción que figuran en la tabla 3, de los que para empezar practicando se recomienda utilizar i (insertar texto a

Si el carácter de escape se encuentra en la cadena, se deberá "escapar del carácter de escape"

partir de la posición en la que se encuentra el cursor), x (borrar el carácter que se encuentra bajo el cursor) y o (crear una nueva línea siguiente a aquella en la que se encuentra el cursor, y pasar sobre ella a modo inserción). Con estos tres comandos y los cuatro movimientos anteriormente indicados se podrá editar cualquier texto y, con la práctica, se irán aprendiendo el resto.

TABLA 2

EXPRESIONES REGULARES EN VI

.	
*	
.* [xyz]	
[x-z]	Cualquier carácter
Repetición (0 ó más)	
Cualquier cadena	
Enumeración	
Rango de caracteres	[a-z0-9]
[^A-Z]	
^	
$	
\	Rangos
Exceptuando	
Principio de línea	
Fin de línea	
Escape carácter	

En el modo inserción todo lo que el usuario teclea es incorporado al texto, hasta que vuelve al modo comando pulsando la tecla ESC. Hay que recordar que cada comando puede ir precedido de un número entero, que indica el número de veces que se debe realizar. Así, si por ejemplo se desea copiar un bloque de texto de 5 líneas, bastará colocar el cursor sobre la primera de ellas y teclear 5Y, con lo que quedarán grabadas en un buffer. Después se colocará el cursor sobre la línea anterior a dónde se quiera copiar el bloque, y se pulsará p para vaciar el buffer.

OTRAS ACCIONES CON VI

Existen otras acciones relativas a objetos (tabla 4) que se realizan en modo comando. Tras pulsar la tecla correspondiente a la acción, habrá que indicar la tecla correspondiente al objeto sobre el que se desea realizarla (que coincide con las teclas de movimientos). Así por ejemplo:

```
dd borra la línea sobre la que se encuentra el cursor
dw borra la palabra sobre la que se encuentra el cursor
dG borra hasta el final de texto
d} borra hasta el párrafo siguiente
d$ borra desde el cursor hasta fin de línea
c$ cambia desde el cursor hasta fin de línea
cw cambia desde el cursor hasta fin de palabra
y$ almacena en un buffer desde cursor hasta fin de línea
```

En la acción c, se pasa a modo inserción automáticamente. Para delimitar el objeto que se está cambiando, aparecerá un signo $ al final del mismo (fin de palabra, de línea...). Una vez se haya terminado la edición, se indicará del modo habitual la vuelta al modo comando pulsando ESC. Cada vez que se utilice la acción d para borrar, el objeto eliminado pasará a un buffer del mismo modo que si hubiera empleado la opción y. De esta forma, se podrá recuperar con p o P, lo que es útil para realizar movimientos de bloques.

Del mismo modo que ocurría con talk y que ocurre con otros muchos comandos, mediante la pulsación de ^L se redibuja la pantalla, opción muy útil

TABLA 3

EDICIÓN E INSERCIÓN

a	
i	
o	
r	
x	
u	
p	
Y	
~	Añadir tras cursor
Insertar en cursor	
Añadir línea siguiente	
Reemplazar carácter	
Borrar bajo cursor	
Undo última acción	
Pegar después	
Grabar línea	
Cambiar MAYS/min	A
I	
O	
R	
X	
U	
P	
.	
J	Añadir a fin de línea
Insertar al principio	
Añadir línea anterior	
Sobreescribir	
Borrar antes de cursor	
Undo acción en línea	
Pegar antes	
Repetir último comando	
Juntar línea siguiente	

en casos en que otro proceso ha escrito sobre el mismo terminal, ya que permite volver a mostrar el contenido del buffer del editor.

Si en modo comando se pulsa !!, se podrá teclear un comando que será ejecutado por una shell. El resultado (la salida estándar) de este comando se incorporará al texto a partir de la posición en que se encuentra el cursor. Si, en cambio, se teclea :!, también se podrá teclear un comando que será también ejecutado por una shell, pero esta vez el resultado sólo aparecerá en pantalla sin quedar incorporado al texto. Para volver a vi será necesario pulsar Intro.

COMANDOS INVOCADOS POR ":" Y MACROS

Además de la opción :! comentada, existen otros comandos invocados mediante la pulsación del carácter "dos puntos" : en modo comando. Se trata de órdenes complejas que precisan de más de una letra para ser interpretadas. Por ello, al pulsar ":" el cursor pasa a la última línea del terminal, que es una línea de status, en la que se podrán escribir y editar los comandos más complejos que no serán validados hasta que se pulse Intro (tabla 5). Como indica la tabla, para realizar sustituciones, se pulsará

:s/<texto original>/<nuevo texto>

Si la cadena a sustituir aparece varias veces en una línea, se sustituirá únicamente en la primera aparición. Para indicar una sustitución total se debería escribir

:s/<texto original>/<nuevo texto>/g

Del mismo modo pueden indicarse rangos de líneas de la forma:

:primera,última/<texto original>/<nuevo texto>

en estos rangos, la línea actual se representa por el carácter punto (.) y la última, por el carácter dolar ($).

Para salir del editor, se indica :q, pero si se han producido cambios en el texto, el programa avisará y evitará salir sin salvar. Por esta razón se indicó anteriormente :q! como la forma de salir. Se emplea el carácter "!" tras :q o :w para indicar a vi que realice la acción sin más miramientos. Como también indica la tabla 5, los comandos :w (escribir) y :q (salir) pueden ser agrupados en :wq, que tiene un comando equivalente en ZZ.

La distribución de Linux contiene editores como el ed, sed, vi, emacs y joe

PROBLEMAS DE TRANSPARENCIA

Un problema de transparencia es aquel que se produce en un canal de comunicación, o sistema de representación de la información, cuando secuencias de control de los datos pueden formar parte de la propia información. En vi puede surgir un problema de transparencia de la necesidad de incluir en el buffer caracteres especiales que no pueden ser escritos por tratarse de caracteres de control, como por ejemplo ^H, que borraría un carácter en lugar de incluirse en el buffer; ^C o ESC que terminarían una inserción, etc. En estos casos, habrá que teclear ^V previamente para indicar al editor, que el siguiente carácter debe ser introducido en el buffer de forma literal en lugar de ser interpretado. Esto es especialmente importante a la hora de editar ficheros de configuración del sistema, como el .login, donde se puede indicar que caracteres deben ser interpretados a nivel de shell como caracteres de borrado, interrupción, etc.

Un ejemplo muy útil de la utilización de esta característica es la conversión de ficheros ASCII de formato DOS a UNIX y viceversa. El sistema DOS coloca los caracteres LF (Line Feed o avance de línea) y CR (Carriage Return o retorno de carro) al final de cada línea de texto, mientras que UNIX no utiliza el retorno de carro. Por ello, si se observa un fichero de texto ASCII creado en UNIX desde un entorno DOS, todas las líneas comenzarán donde terminó la anterior, por lo que el texto aparecerá completamente descolocado. Se puede convertir un fichero ASCII de DOS a UNIX editándolo con el vi y tecleando

:1,$s/^V^M/

es decir: "desde la línea primera hasta la última, sustituir el carácter de

TABLA 4

OTRAS ACCIONES

d	borrar objeto	c	Cambiar objeto
y	copiar objeto	!	filtrar objeto

retorno de carro por... ¡nada!". El proceso inverso se podría realizar con

:1,$s/$/^M

o lo que es lo mismo: "desde la línea primera hasta la última, sustituir fin de línea por retorno de carro". Como se puede apreciar si se leen las órdenes sin tener cierto conocimiento de vi, éste no se caracteriza por ser precisamente intuitivo.

En caso de que el problema de transparencia se produzca en un comando de sustitución (porque el carácter "/" se encuentre entre los que forman el texto original o nuevo), vi lo soluciona considerando delimitador al primer carácter tras la letra "s". Así, para la sustitución del apartado anterior, se podría también haber tecleado

:s?<texto original>?<nuevo texto>

Un tercer problema de transparencia puede surgir, de la necesidad de buscar en una cadena, que contenga alguno de los caracteres utilizados para especificar patrones de búsqueda. Por ejemplo, en un programa C, podrá interesar localizar un comentario /*, donde aunque el problema del carácter "/" como delimitador puede ser solventado, el del carácter "*", como indicador de repetición (0 ó +) estropearía la búsqueda. Del mismo modo se podría tratar de buscar una referencia bibliográfica [Klir], que sería interpretada como una enumeración. Para ello, se puede utilizar el carácter de escape "\" que elimina la función especial del carácter que le sigue de forma inmediata, obligando a que éste sea interpretado de forma literal. Así, las búsquedas anteriores podrán ser indicadas como:

/\/*
/\/[Klir\]

aquí de nuevo surge un problema típico de transparencia, el de que el carácter de escape forme parte del texto, por ejemplo, en un texto que contenga una ruta DOS: c:\utilidad\wp51. Para solventarlo se utiliza la solución también típica de "escapar del carácter de escape":

/c:\\utilidad\\wp51

MACROS

Un comando muy útil que permite definir macros en vi, es la orden :map. Su sintaxis es simple: :map <tecla> <secuencia de órdenes>. Eso sí, habrá que tener cuidado con los problemas de transparencia. Así, por ejemplo:

:map v /vi^VESC

donde la cadena ESC representa la tecla de escape, definirá una macro que se activará al pulsar v cuya acción consistirá en localizar la cadena vi en el buffer.

Mediante ":" se pueden introducir órdenes complejas que requieren más de una letra

Otra macro más interesante podría ser:
:map g bi"^VESCea"^VESC

que entrecomillaría la palabra sobre la que se encuentra situado el cursor al pulsar g en modo comando.

La razón de haber elegido estos nombres tan "esotéricos" para las macros es que éstas deben ser de un sólo carácter, y conviene que no sean nombres de comandos vi.

TABLA 5

COMANDOS COMPLEJOS	
:s/xx/yy	
:s/xx/yy/g	
.,$/xx/yy	
:q	
:q!	
:w	
:w fichero	
:wq	
:r fichero	
:! comando	Sustitución en línea
Sustitución global en línea	
Sustitución en rango de líneas	
Salir	
Salir por fuerza	
Grabar cambios	
Grabar en "fichero"	
Grabar y salir (ZZ)	
Leer fichero	
Ejecutar comando	

Como nombre de tecla también es posible utilizar #número para aludir a una tecla de función.

CONFIGURACIÓN DEL EDITOR VI

El editor vi incluye un conjunto de opciones, que pueden ser modificadas para configurar su comportamiento. Al igual que ocurría con el mail, existen varios tipos de opciones, en este caso tres, que pueden ser clasificadas en numéricas, de cadena y booleanas. Para dar valor a una opción numérica o de cadena se utiliza el comando :set opción=valor mientras que las booleanas pueden ser activadas mediante :set opción o desactivadas mediante :set noopción. Para ver el valor de las opciones en un determinado momento se puede teclear:

:set all muestra el valor de todas las opciones

:set opcion? muestra el valor de la opción indicada

:set muestra los valores que no son por defecto

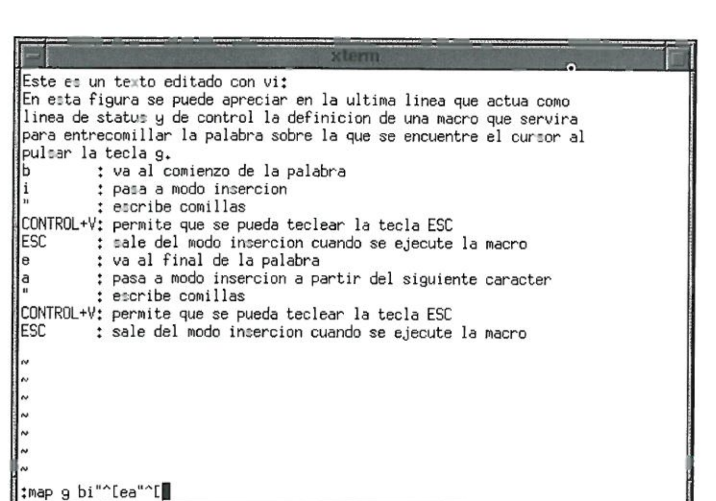
Entre las opciones más útiles se encuentran:

ai: AutoIndent. Activa la indentación (sangrado) automática, que consiste en que, cada nueva línea que se introduce comienza automáticamente donde comenzó la anterior. Para rechazar una indentación basta teclear ^D en cuanto ésta se produce. Por defecto, se encuentra desactivada.

ic: IgnoreCase. Cuando se encuentra activada provoca que vi no diferencie mayúsculas y minúsculas en las búsquedas. Por defecto esta desactivada, con lo que vi será case sensitive.

nu: NUmber. Cuando se encuentra activada causa que vi muestre los números de las líneas. Por defecto no lo está.

sm: ShowMatch. provoca que cada vez que se escribe (en modo inserción) un carácter ")" o "}", vi desplace el cursor hasta el carácter abierto que



```
Este es un texto editado con vi:
En esta figura se puede apreciar en la ultima linea que actua como
linea de status y de control la definicion de una macro que servira
para entrecomillar la palabra sobre la que se encuentre el cursor al
pulsar la tecla g.
b         : va al comienzo de la palabra
i         : pasa a modo insercion
"         : escribe comillas
CONTROL+V: permite que se pueda teclear la tecla ESC
ESC       : sale del modo insercion cuando se ejecute la macro
e         : va al final de la palabra
a         : pasa a modo insercion a partir del siguiente caracter
"         : escribe comillas
CONTROL+V: permite que se pueda teclear la tecla ESC
ESC       : sale del modo insercion cuando se ejecute la macro

~
~
~
~
~
~
~
:map g bi"^[ea"^[
```

Figura 1: Aspecto que presenta el editor vi al arrancar.

le corresponde, opción muy útil en programación.

Existe una variable de entorno de shell que puede almacenar los valores de estas variables llamada EXINIT. Al arrancar vi, pregunta al S.O. por el valor de esta variable. Por ello, es útil definirla en el fichero .login para las shells de la familia csh o en el .profile para las shells de la familia sh (se verá la función de estos ficheros en el capítulo dedicado a shells). Para, por ejemplo, activar las opciones ai, sm e ic, en el primero de los casos (csh) habrá que incluir la línea:

```
root::0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:/bin/bash
games:*:12:100:games:/usr/games:
man:*:13:15:man:/usr/man:
postmaster:*:14:12:postmaster:/var/spool/mail:/bin/bash
ftp:*:404:1::/home/ftp:/bin/bash
echeva:JeiVkB7pCMaws:501:100:Fernando J. Echevarrieta:/home/echeva:/bin/bash
gonzo::418:100::/home/gonzo:/bin/bash
satan::419:100::/home/hell:/bin/bash
snake::420:100::/home/pit:/bin/bash
sprog:M1VDtQW7MYWys:502:100:Solo Programadores:/home/sprog:/bin/bash
~
~
"/etc/passwd" [READONLY] 21 lines, 789 chars
```

Figura 2. La última línea sirve para introducir comandos complejos.

setenv EXINIT 'set ai sm sc'
y en el segundo (sh), las líneas:
EXINIT='set ai sm sc'
export EXINIT

NOTA: Linux, por defecto, genera las cuentas con la shell bash, la cual es de la familia sh.

Al igual que con el programa mail y tantos otros, también se podrá realizar la configuración del editor a través de un fichero: .exrc, que debe encontrarse en el directorio de usuario. Es en este fichero, donde suelen indicarse las asignaciones de opciones y las definiciones de macros.

Es extremadamente desaconsejable modificar ficheros como root si no se sabe EXACTAMENTE lo que se está haciendo

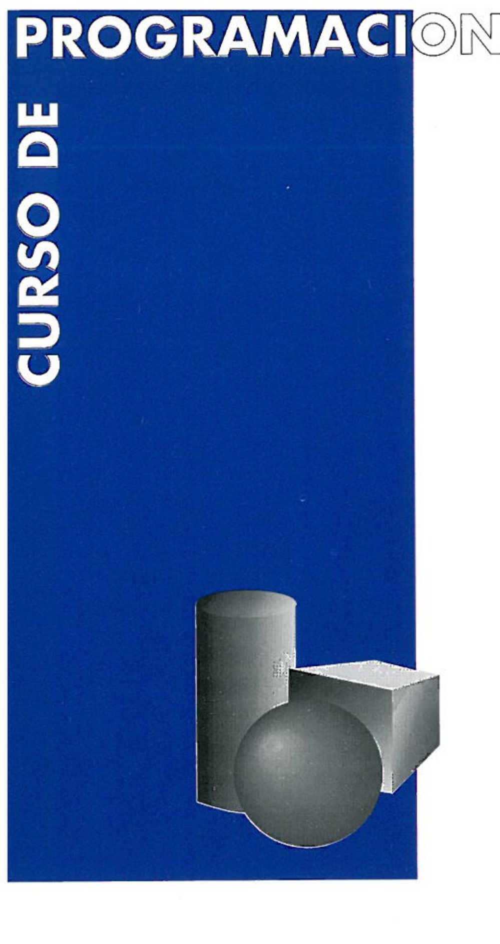
POR ÚLTIMO

Es un buen ejercicio para el lector que practique modificando algunos ficheros, como los indicados en este y otros capítulos: .login, .profile, .mailrc, .exrc, pero es extremadamente desaconsejable que modifique ficheros como root del sistema, ya que, un error pude conducir a un desastre, que implique una completa reinstalación. A aquellos lectores que hayan instalado Linux en sus máquinas, y no hayan creado aún ninguna cuenta, se les recomienda encarecidamente que lo hagan para practicar desde la misma. Para ello pueden utilizar el comando adduser explicado en el capítulo tercero del curso.

Es importante destacar que lo hasta ahora expuesto, es una simple y sucinta introducción a vi, sobre el que sería necesaria una serie completa, para explicarlo en toda su potencia y extensión. Las tablas contienen los movimientos, comandos, e información general más usuales, y que todo usuario con experiencia en vi conoce de memoria, pero todas están incompletas y podrían ser ampliadas, aunque se ha considerado que esto reduciría su capacidad didáctica, y de consulta rápida. Para comenzar a manejar vi es recomendable emplear únicamente los cuatro movimientos, y las tres instrucciones que se indicaban al comienzo, y más tarde ir incorporando nuevas acciones.

EL PRÓXIMO CAPÍTULO

En el próximo capítulo se estudiará un tipo especial de comandos llamados filtros, que ya fueron adelantados en el capítulo tres. Con un editor como vi se podrán generar nuevos comandos UNIX de gran potencia, interconectando estos filtros entre sí en un fichero ejecutable. Con ellos, y con el procesador de patrones awk, se dispondrá de una formidable plataforma de herramientas que alcanzará plena potencia, cuando se le incorporen las complejas facilidades de sustitución de las shells como intérpretes de comandos, y las sencillas estructuras de control de las mismas como lenguajes interpretados. ■



TERMINANDO EL CUADRO

Ignacio Cea

En esta entrega va usted a aprender a exprimir un poco más los recursos que le proporciona el lenguaje C++. Le hará ver también la utilidad del reuso de software con sus propios ojos. Hoy aprenderá, por tanto, que es el C++.

Es muy normal que todo aquel que se inicie dentro del mundo del C++ piense, en un principio, que el tema del reuso del software es algo tan utópico como podría serlo en cualquier otro lenguaje. Este artículo pretende, básicamente, convencerle de la diferencia; sin embargo, también aprovechará la ocasión para dar las últimas pinceladas a las características C++, que han ido siendo desglosadas a lo largo de todos estos meses.

Para ello vamos a realizar una aplicación que suele ser crítica en tiempo y recursos. Vamos a redireccionar la rutina del reloj con el fin de, que una serie de letras oscilen dentro de la pantalla texto. Al final se le propondrá que adapte esa aplicación para que, por ejemplo, además de las letras se muevan rectángulos. Caerá entonces en la cuenta de que esa tarea podría llevarle no más de 10 minutos.

CONSTANTES

Las constantes tienen una relevancia especial dentro del C++. Constantes pueden ser tanto los atributos definidos dentro de las clases, como los métodos que los manejan, como los objetos construidos a partir de ellas.

En C++ no es aconsejable el empleo de macros (define) de una forma tan asidua a como se hacía, por ejemplo con su "hermano pequeño", el C. Cuando se desarrolla un proyecto tanto en C como en C++ no es extraño encontrar dos personas que desarrollando partes distintas elijan las mismas macros para representar conceptos diferentes. Así por ejemplo MAXLONGITUD puede hacer referencia bien a la máxima longitud de una cadena de caracteres, bien al máximo tamaño de una mesa.

Una de las características fundamentales del C++ es, como ya se ha repetido hasta la saciedad, la encapsulación. Las constantes no dejan de ser, en su mayor parte, algo asignable directamente a una clase y por tanto deberían ser definidas dentro de aquellas.

Observe, como ejemplo, el fichero "ALARMA.HH". El máximo número de alarmas está definido dentro de la clase Alarma como un miembro estático constante. Su valor es asignado dentro del fichero "ALARMA.CPP", que especifica el cuerpo de los métodos de la clase.

ELEMENTOS CONSTANTES

En C++ pueden ser constantes, tanto los atributos (estáticos o no) de una clase, como los métodos que los emplean (no estáticos), como los objetos inicializados. Ni que decir tiene que pueden estar situados en cualquier parte de la declaración de la interfaz de una clase; es decir, tanto en la parte pública, como en la privada, o en la protegida.

Sin duda que usted tiene perfectamente claro, qué significa que un atributo de una clase sea constante: su valor no puede ser alterado por nadie. Pero, ¿qué significa que un método no estático sea constante?

ATRIBUTOS CONSTANTES

En C++ los atributos no estáticos constantes se declaran dentro de la definición de la estructura de la clase y reciben su valor dentro de los constructores de la clase.

Los atributos estáticos, por otro lado, también se declaran dentro de la estructura de la clase pero su valor se inicializa como si se tratara de cualquier otro atributo estático; es decir, en la parte superior del fichero de decla-

ración del cuerpo de los métodos, normalmente;

OBJETOS MIEMBRO CONSTANTES

Un objeto constante es lo mismo que cualquier otro atributo de una clase; es decir, si es estático deben recibir su valor inicial en la parte superior del fichero ".CPP", mientras que si no lo es debe ser inicializado dentro del constructor de la clase que lo contenga.

Sin embargo con los objetos constantes inmediatamente surge una gran incongruencia: imagine que dentro de la clase de ese objeto constante se ha definido un método que altera el valor interno de cualquiera de los atributos del objeto. Si esto es así, ¿tiene sentido tener un objeto constante cuyo valor interno podría ser variado tan fácilmente?. Para evitar esto C++ pone una nueva regla: "Un objeto constante sólo puede invocar a aquellos métodos que hallan sido declarados también como constantes".

MÉTODOS CONSTANTES

Ya sabe lo que es un método const: "aquel que no altera el valor de los atributos del objeto". Para declarar un método constante debe colocar la palabra const al final de la declaración del método. Esa palabra clave debe repetirla dentro del fichero que defina el cuerpo del mismo. Sólo pueden ser métodos constantes los métodos member de la misma; es decir no pueden serlo ni las funciones friend, ni las estáticas.

Como ejemplo fíjese en la sobrecarga de operadores definida dentro del fichero "TIEMPO.HH". El método "+" puede, tanto definirse como constante, como recibir la referencia constante a un objeto ya que, por un lado, el objeto recibido no va a modificarse (es sólo un dato de entrada) y, por otro, el método no va a modificar los atributos del objeto invocador. Sin embargo, el método "+=" puede seguir recibiendo la referencia constante, pero no puede ser declarado como constante, pues va a alterar el valor del objeto que lo llama (recuerde que el objeto se actualizará con el resultado de la suma). El operador "<<" no puede ser declarado como const pues es de tipo friend.

CUANDO SE TIENEN DOS PADRES

C++ es uno de los pocos lenguajes que permiten la herencia múltiple (SmallTalk, por ejemplo, no). Es decir, puede llegar a existir una clase que herede de varias. La mayor parte de las metodologías de trabajo orientadas a objetos no recomiendan esa situación e, incluso, algunos programas de control de la calidad del software avisan de que ese manejo no es algo muy "ético" (QAC++, por ejemplo, así lo hace). Sin embargo, en algunas situaciones puede llegar a ser muy útil, e incluso las mismas librerías estándar del C++ hacen uso de este recurso (la estructura de clases stream, por ejemplo). Por este motivo es conveniente que conozca la forma en la que opera, y los problemas a los que tendrá que enfrentarse.

La mejor manera de que comprenda porqué puede llegar a darse esta situación es con un ejemplo: imagine que una librería de clases que usted compra tiene ya definidas, por separado, las clases "sofá" y "cama". Suponga, por otro lado, que usted necesita crear una clase que representa las características de los sofascama. ¿Qué mejor manera que crear esta nueva clase que herede de las otras dos?.

LA FORMA DE HERENCIA

Para conseguir una clase con herencia múltiple basta con colocar, tras la declaración de la clase y separadas por comas, los nombres de todas las clases padre de la nueva:

```
class CLASEHIJA : public A,
        public B, private C
        {...
        };
```

Delante del nombre de cada clase ha de colocar el identificador de acceso relativo a esa herencia. El hecho de que la mayor parte de las veces ese identificador sea public, no quiere decir que cualquier otro no pueda emplearse. Observe la figura 1 para comprender el alcance de las distintas formas de herencia.

	EVOLUCION DE MIEMBROS		
HERENCIA	**PUBLICOS**	**PROTEGIDOS**	**PRIVADOS**
PUBLICA	PUBLICOS	PROTEGIDOS	PRIVADOS
PROTEGIDA	PROTEGIDOS	PRIVADOS	PRIVADOS
PRIVADA	PRIVADOS	PRIVADOS	PRIVADOS

Figura 1.

PROBLEMAS

La herencia múltiple puede llegar a plantear ciertas incongruencias. Supóngase que su nueva clase hereda de forma pública otras dos. Imagine, también, que en cada una de esas clases existen métodos que se llaman de igual manera (eso es, recuerde, debido al polimorfismo). Si, ahora, crea un objeto de la nueva clase e intenta invocar a alguno de esos métodos repetidos, ¿a cuál de los dos llamará el compilador?.

El compilador no es tan inteligente, y ante una situación así visualizará un error. Usted ha de ser el encargado de resolver la ambigüedad. Una forma de hacerlo es, por ejemplo, colocar, antes de la llamada al método, el nombre de la clase donde buscarlo seguido por el operador de entorno (::) y el identificador de la función en cuestión. Observe la figura 2.

Si, por otro lado, dentro de la función hija se declara un método con el mismo identificador que los de las clases padre, el compilador no encontrará ambigüedad alguna. Observe la figura 3. En este caso sólo si quisiera emplear un método distinto al que el compilador elige debería hacer uso del operador de ambiente antes mencionado.

REUSO EN HERENCIA

La herencia es la carta de presentación del reuso. Cada nueva clase creada suele poder emplear todos los atributos y miembros de las clases padre. En definitiva, para conseguir, una alta funcionalidad en sus clases no es necesario, normalmente, que escriba grandes cantidades de código, pues la mayor parte puede estar ya definido dentro de sus padres.

Su librería de clases irá incrementándose conforme desarrolle más y más proyectos, y el tiempo que empleará en desarrollarlos será cada vez menor, pues disminuirá el código escrito gracias al empleo del reuso.

```
class A{
...
METODO (void);
...
};
class B {
METODO (void);
...
};
class C: public A, public B {
};
... ¿Qué método METODO debo usar al
llamarlo?
void main (void)
} C obj;
obj. METODO ()       ——►  AMBIGÜEDAD
obj. A:: METODO();  }
obj. B:: METODO();  }     Solución
```

Figura 2. Resolviendo ambigüedades.

No se ilusione. Para conseguir un alto grado de reuso, hay que analizar, diseñar y programar pensando en él. De lo contrario puede llegar a tener una aplicación altamente estructurada, encapsulada, fácilmente extensible, etc... pero de la que pueda emplear pocas cosas de cara a otro sistema.

REUSO

Observe las clases tiempo, pixel, alarma, letra, etc. que vienen con esta entrega. Pueden emplearse en infinidad de aplicaciones diferentes y llegar a ser tan útiles como cualquier otro tipo de dato estándar de C++. Son muy reusables, pues no están ligadas a nada ni a nadie, igual que un int.

Sin embargo, hay un detalle que, incluso, se hace patente en el ejemplo de este artículo. Buena parte de la funcionalidad puesta en juego, jamás se emplea. No existe ningún lugar de la aplicación donde se emplee el operador == de pixel, o el de alarma, o el de letra. Pensar en el reuso sin embargo, obliga a ponerlas, pues otras aplicaciones si que pueden emplearlas.

Tampoco es necesario exagerar. Evidentemente, la funcionalidad que puede tener una clase, incluso aunque sea básica, es enorme. Sin embargo, ponerla toda en funcionamiento, incrementaría notablemente el tamaño de cualquier aplicación. Lo ideal es dar sólo la funcionalidad básica, y esperar que el reuso a través de la herencia la vaya ampliando. Claro está que esto, es una opinión personal fruto de la experiencia y cada cual puede entender cosas distintas.

EL TAMAÑO DE LOS EJECUTABLES

Los ejecutables en C++, suelen tener inicialmente un tamaño más grande que los desarrollados en otros lenguajes. Prueba de ello es, el típico programa "Hola mundo" (figura 4). En ese ejemplo se emplea el objeto cout. Ese objeto pertenece a una clase (ostream_withassign), cuya funcionalidad es enorme, y de la que usted sólo emplea una mínima parte.

Sin embargo, conforme crece el tamaño de la aplicación, el reuso y la encapsulación evitan repetir código, y el número de bytes empleados disminuye por línea de código escrita.

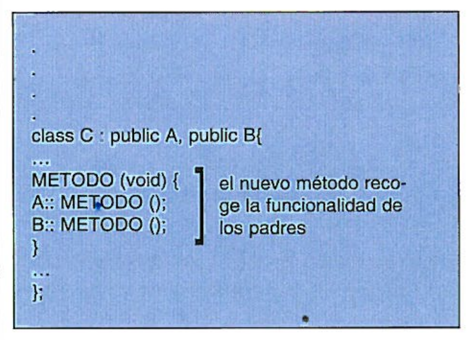


Figura 3. No hay ambigüedad.

Los expertos coinciden en que existe un tamaño óptimo, a partir del cual los programas C++ son rentables también en tamaño (véase figura 5). Los estudios citados tienen valores concretos cuya exposición no es fin del presente artículo.

LA VELOCIDAD DE EJECUCIÓN

El empleo del reuso produce un aumento en el número de llamadas entre funciones (métodos en C++), lo cual redunda en una pérdida de rendimiento general de las aplicaciones. El empleo correcto, sobretodo de los métodos inline, y de las referencias, hace que tal pérdida sea prácticamente nula, como lo muestra el hecho de que pueda ser empleado para una aplicación tan crítica, como la de este mes que, incluso abusa de tales llamadas.

De todas formas, hay que hacer notar que la velocidad de ejecución no es una de las razones básicas para elegir C++ como lenguaje de desarrollo. Si lo son, por el contrario, tanto la fácil extensibilidad y mantenimiento, como la claridad.

LA APLICACIÓN EJEMPLO

SÓLO PROGRAMADORES proporciona este mes un ejemplo construido con clases totalmente reusables (figura 6). En este ejemplo se han reunido todas las características expuestas durante estos meses. Básicamente, se trata de dar al usuario la posibilidad de definir rutinas, que puedan ser ejecutadas periódicamente en el tiempo. Como ejemplo de ello, se enseña a hacer parpadear 2 letras en las esquinas de la pantalla. La aplicación sólo podrá ejecutarse en máquinas DOS.

COMO COMPILAR LOS EJEMPLOS

Compilar el ejemplo de esta entrega es sensiblemente más complicado que los de meses anteriores. Además de cargar el proyecto, hay que asegurarse de que la detección de desbordamiento de pila, y el empleo de variables register estén desconectados. En el compilador BorlandC++, las opciones están, respectivamente, en los menús linker y C++ optimizations. Tras esto, lo único que hay que hacer es ejecutarlo.

```
# include <fortream.h>
Void main (void)
{
cout << "Hola mundo" <<endl;
}

# include <stdio.h>
void main (void)
{
printf ("Hola mundo\n");
}
```

Figura 4. Hola mundo.

El ejecutable que incluye el disco está creado para máquinas 386 o superiores, por lo que si su ordenador no responde a estas características, deberá recompilar el ejecutable.

PARÁMETROS GENÉRICOS

Dentro de C++ existe la posibilidad de establecer que los atributos que un

método va a recibir, que pueden ser cualesquiera, tanto en número, como en tipo (siempre y cuando sean estándares).

Para indicar que los parámetros de un método o función pueden ser cualquiera, basta colocar tres puntos en la declaración del prototipo. En este recurso se basan funciones como las del grupo de printf.

Los métodos para el manejo de interrupciones, reciben como parámetros los valores de muchos de los registros del microprocesador. Sin embargo, es absurdo tenerlos en cuenta cuando aquellos no van a ser empleados para nada. Tal es la situación de la rutina de interrupción de nuestro ejemplo. Existen muchas funciones para poder descifrar los parámetros pasados a un método de estas características.

OTRA FORMA DE USAR LOS PARÁMETROS

Suponga dos clases, una hija de la otra, que redefinen el empleo del operador "=". Es normal, aunque no obligatorio como con cualquier otro método, que dentro de la clase hija se intente emplear el operador "=" de la otra para evitar reescribir código (reuso).

Para ello debe existir alguna forma de invocarlo. Existen varias. Una de ellas y, ya que se trata de un método cualquiera, es invocarlo directamente por su nombre:

```
class A {
  ...
  A &operator = (const A &);
  ...
};

class B : public A {
  ...
  inline B &operator = (const B &)
  { A::operator = (B); . . .}
};
```

La única diferencia es que el nombre de ese método es compuesto (operator =). Por lo demás todo es igual. Si no pusiéramos A:: el ejecutado volvería a ser el mismo método, y nos encontraríamos ante una recursividad.

OPERADORES NO DECLARADOS

Si dentro de una clase hija no se redefine el empleo de un operador cualquiera, y el compilador se encuentra una línea de código donde se pretende emplearlo, buscará por la línea jerárquica hasta encontrar uno. Si la clase tiene varias líneas jerárquicas, y en todas ellas se encuentra la definición de ese operador, nos mostrará el mensaje de error de ambigüedad al que en párrafos anteriores se hizo mención. Si sólo se encuentra una línea jerárquica, el error no aparecerá.

Tal es el caso de las clases hijas de alarma en el ejemplo. Nunca es declarado en ninguna de ellas el operador ">=". Sin embargo, se puede emplear,

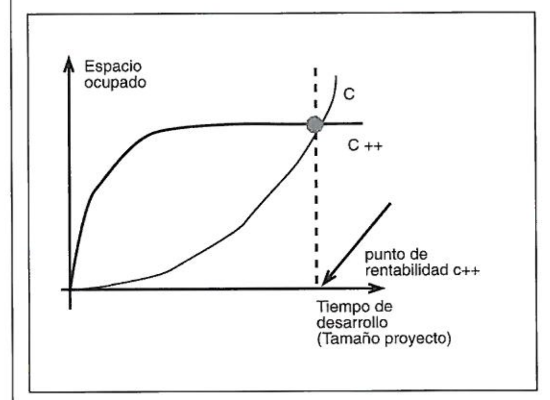


Figura 5. Evolución del tamaño.

ya que la clase tiempo, padre de todas, si que lo contempla.

LA LETRA DINÁMICA. PENSANDO EN EL REUSO

Las clases Pixel, Tiempo, Alarma y AlarmaPeriodica, constituyen la base del ejemplo, dan la posibilidad de describir cualquier rutina periódica.

Nuestro propósito final es, hacer que varias letras oscilen de forma periódica dentro de la pantalla. Es claro que nuestra nueva clase debe heredar de AlarmaPeriodica, que es quien le proporcionará la característica de alternancia en el tiempo. No obstante, vamos a tratar con algo más que una letra pues tendrá un color y una posición. Características que se van a considerar suficientes para generar una nueva clase.

La clase Letra es algo completamente reusable en múltiples aplicaciones. La clase LetraPeriodica, por otro lado, aúna las características de una alarma y de una letra, de ahí que halla sido creada como hija de ambas. Los objetos de la aplicación son instancias finales de ésta. Hubiera sido fácil crear la letra dinámica heredando directamente de AlarmaPeriodica, pero es muy probable que, si así se hubiera hecho, hubiera que reescribir el código de la clase Letra en alguna otra aplicación que lo necesitara. De esta forma tenemos la misma funcionalidad y mucho más reuso.

UN RECTÁNGULO DINÁMICO

Piense ahora, en ampliar el ejemplo para que un rectángulo aparezca, a una hora en concreto en pantalla y, tras cierto número de segundos, desaparezca. En primer lugar, la forma de comportamiento ya no responde a la de una alarma periódica, que cada cierto lapso de tiempo se activa, sino más bien a la de un despertador.

Necesitamos crear una nueva clase, llamada por ejemplo Despertador, que herede de Alarma (hemos reusado todo el código de la alarma). Hemos de crear por otro lado la clase Rectángulo que recoge el comportamiento de esta figura geométrica, y que bien puede ser la creada para los ejemplos del artículo 3. Por último, el RectanguloPeriodico surge de las dos, igual que LetraPeriodica surgía de Letra y AlarmaPeriodica.

UN CONSEJO

Tenga en cuenta que la rutina de reloj se ejecuta unas 12 veces por segundo. Si el tiempo que el microprocesador invierte en ejecutar sus requerimientos, supera ese tiempo pueden producirse resultados imprevisibles (normalmente el colgamiento de la máquina).

PRÓXIMA ENTREGA

En la próxima entrega concluiremos el repaso general, y empezaremos a ahondar en la forma de resolver algunos, de los problemas que el C++ no soluciona. El primero de ellos será abordar la integridad relacional. ■

Linux PC

INSTALACION DEL SLACKWARE 2.1.0

David Aparicio

Este artículo no pretende recorrer exhaustivamente toda la instalación, y supone cierto conocimiento de lo que se desea instalar. Se recomienda repasar la documentación, tanto en el CD como los programas de ejemplo del disquete, antes de proceder a la instalación.

INSTALACIÓN DE LINUX: PASO A PASO.

Aquellos que se desenvuelvan sin problemas leyendo del inglés, pueden consultar los pasos de la instalación de los siguientes ficheros:

\linux210\readme.ins (abreviada)
\linux210\install.txt (completa)

Al visualizar algunos textos desde MS-DOS, el contenido parece estar "apelotonado" en unas pocas líneas. Esto sucede porque, en Unix, se emplea un sólo carácter (LF) para marcar un fin de línea, mientras que MS-DOS requiere dos caracteres (CR+LF). El usuario, en MS-DOS, puede examinar el contenido de estos ficheros mediante el comando edit, que no requiere de ambos caracteres.

En Unix, el texto procedente de MS-DOS se visualiza con caracteres de control (^M) al final de cada línea.

Por otra parte, se puede observar que la mayor parte de los ficheros tienen la terminación .gz (comprimido gzip) o .tgz. (tar, comprimido gzip). Para descomprimir, teclear:

```
gzip -d <nombre>
```

Los ficheros .tgz tendrán ahora la terminación .tar. Los ficheros .gz ahora están listos para ser leídos o usados. El formato "tar" agrupa una estructura de subdirectorios y ficheros en un archivo. El comando para ver su contenido es:

```
tar tvf <nombre.tar>
```

El comando para expandir a partir del directorio actual es:

```
tar xvf <nombre.tar>
```

OBTENCIÓN DE LOS DISCOS DE ARRANQUE

Se necesita, primero, obtener un disco de arranque (boot) y otro de instalación (root). El subdirectorio que contiene los ficheros y los programas necesarios es el \linux210\bootdisk. En la documentación en inglés, hay referencias a otros directorios (./bootdsks.144 ./bootdsks.12 ./rootdsks.144 y ./rootdsks.12) que, en la instalación que nos ocupa, son sustituidos por éste.

1) El único disco root disponible es el color144.gz

2) En cuanto al disco de arranque, podemos elegir el que se adecue al hardware disponible. Acuda a la tabla de la página 81 del número con el que se distribuyó el CD. Por ejemplo, suponiendo que se instale con un CD-ROM Panasonic (Creative), el disco sería el sbpcd.gz

3) Una vez elegidos los discos, se descomprimen (gzip -d) en el disco duro.

4) El fichero resultante se vuelca a un disquete con formato, mediante el programa rawrite.exe.

PARTICIONE SU DISCO DURO

El disco duro debe tener espacio para albergar la instalación. Son necesarias dos particiones, y unos 130 Mb si se desea una instalación media, con X-Windows (el sistema gráfico de ventanas en Unix). 40 Mb, o menos, puede

Para proseguir con el curso Unix, es aconsejable aplicar los ejemplos que acompañan los artículos y consultar documentación, tanto en el programa man como en ficheros asociados. Para ello, nada mejor que instalar el sistema operativo Linux en nuestro propio PC, a partir del CD-ROM que se distribuyó en el número 6 de SOLO PROGRAMADORES.

ser suficiente para una instalación mínima. Prepare su disco para cumplir los requisitos anteriores, antes de seguir adelante.

Es conveniente tener copia de seguridad de los programas y datos de valor, antes de realizar cualquier operación.

1) Arrancamos con el disco boot. Se nos da un prompt

boot:

Aquí se permite especificar información adicional para que Linux esté informado del hardware que fuese incapaz de autodetectar. Normalmente no es necesario mas que pulsar Intro.

2) Cuando se pida, introducir el disco root. Al cabo de unos segundos, nos aparece un mensaje en inglés, y el prompt:

login:

Introducir root. Estamos dentro de Linux.

3) Ahora vamos a crear una partición de sistema y otra como memoria de intercambio.

3.3.1) Con fdisk, crear una partición de swap (memoria de intercambio), del tipo 82 (hex). En Unix, es corriente seguir la norma de que el swap sea el doble del tamaño de RAM instalada. Para 8 Mb de RAM, 16 Mb de swap. Esta partición añade memoria extra al sistema para operaciones puntuales, y es vital para usuarios con 4 Mb de RAM, como se verá más adelante.

3.3.2) Además, crear otra partición para el propio sistema, tipo 83 (hex), del tamaño que considere conveniente. Se aconseja entre 40 Mb sin X-Windows y 120 Mb para un uso medio.

NOTA: Para nombrar las unidades, Linux utiliza cuatro caracteres. Los dos primeros indican si el disco es IDE (hd) o SCSI (sd). El siguiente carácter indica el número de disco físico ('a' para el primero...).

Por último, un dígito indica el número de la partición de ese disco. Ejemplos:

/dev/hda1 : Primer disco IDE, primera partición.

/dev/sdb : Segunda unidad SCSI, completo.

/dev/sda3 : Tercera partición de la unidad 0 SCSI.

Se puede experimentar libremente con las opciones de fdisk, puesto que los cambios no se almacenan hasta pulsar 'w'. Es posible ignorar los cambios con 'q'. Compruebe el fichero FDISK.TXT que se incluye en el disquete.

4) Si el equipo sobre el que se está instalando posee 8 o más Mb de RAM, puede saltar a la siguiente sección. En caso contrario, si sólo posee 4 Mb de RAM, el programa setup es incapaz de arrancar, y debe activarse previamente la partición de swap:

4.1) Se formatea mediante el comando

mkswap -c <partición_swap> <tamaño_en_Kbytes>

El tamaño_en_Kbytes se puede obtener con el comando "fdisk -l" y observando la quinta columna (Blocks) de la partición de swap.

4.2) Se activa esta memoria extra mediante:

swapon <partición_swap>

A partir de ahora el tamaño de swap se ha añadido a la RAM que hubiese instalada.

Figura 1. Creación de discos y arranque.

EL PROGRAMA INSTALADOR

Tecleando setup (el programa general de instalación), se permiten operaciones como formatear las particiones Linux, seleccionar el teclado, ratón, lector CD-ROM, y elegir los contenidos que se desean instalar.

En la primera instalación, SETUP le permite formatear las particiones creadas previamente mediante fdisk. Cuando aparezca la pantalla referente al la configuración del swap, los usuarios de 4 Mb de RAM no deben volver a usar mkswap ni swapon, puesto que se ha activado esta característica manualmente. Lea detenidamente las instrucciones de pantalla al respecto.

En el apartado de la partición del sistema elegir el tipo ext2. Es el estándar, rápido y fiable, dentro de Linux. El tamaño de i-nodo es una característica que permite elegir el tamaño de bloque

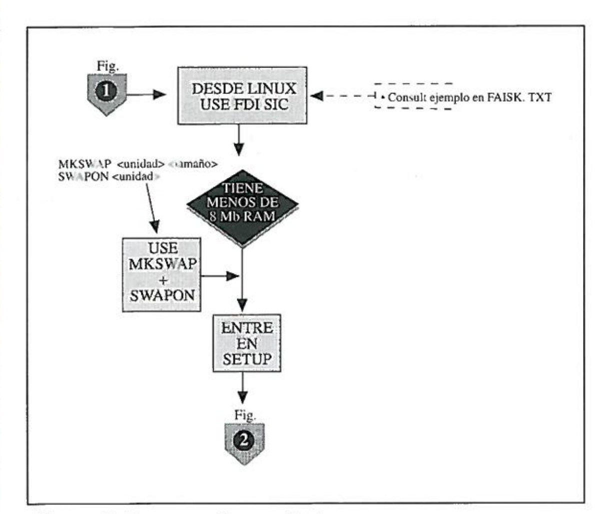


Figura 2. Preparar las particiones.

de su sistema de ficheros. Un buen valor es 2048.

Revisando a grandes trazos las opciones del menú principal, se observa:

K)EYMAP: Permite realizar las operaciones iniciales, como elegir el mapa de teclado. Para el expandido AT en castellano:
/usr/lib/kbd/keytables/es.map

T)ARGET: Indica el directorio a partir del que se van a descomprimir los ficheros a instalar. Dejar tal cual está (/).

S)OURCE: Indica la unidad donde se encuentran los discos originales a instalar. Dado que la distribución ofrecida se encuentra en CD, seleccionar:

5) From CDROM
Elegir el tipo de CD

Seleccionar "Custom", e indicar directorio /linux210

D)ISK SETS: Los discos de Linux se agrupan en series. Cada serie engloba los discos de un tema concreto. Por ejemplo, la serie D (Development) incluye todo lo relacionado con compiladores (C, C++, Pascal , Lisp ..)

Si es su primer contacto con Unix, es mejor instalar sólo la serie A (base) y experimentar un poco antes de lanzarse a una instalación completa, y perderse entre decenas de megas de programas. Esta serie ocupa, una vez instalada, unos 9 Mb de espacio. Otras series con archivos muy útiles son: AP (manuales on-line), D(compiladores), X(X-Windows) y N(Correo, con el lector elm y comunicaciones.). Siempre está a tiempo de invocar setup y añadir nuevas series de discos cuando usted lo desee.

I)NSTALL: Proceda a instalar lo que se ha seleccionado en las opciones anteriores. Se le preguntará por los tagfiles. Estos son los ficheros de configuración para instalar, donde se indica los ficheros que son necesarios, los opcionales, y cuándo preguntar al usuario por ello. Lo corriente es elegir "None" para instalar todo, o "Normal" para que el sistema pregunte.

C)ONFIGURE: Selecciona aspectos concretos de su sistema, como el tipo de ratón, el puerto de su módem, el tipo de CD-ROM o la hora local del sistema (seleccione en este caso "MET", que es la hora de Europa Central). Adicionalmente, puede habilitar el acceso a otras particiones, (MS-DOS o OS/2) directamente desde Linux, para poder leer y escribir ficheros, aunque no para lanzar ejecutables. Lo corriente es tener la partición DOS (si sólo tiene una) en /dosc. Otras particiones se acceden como /dosd, /dose ...

MÉTODOS DE ARRANQUE

Para arrancar Linux, de ahora en adelante, dispone de tres sistemas:

1) LILO, o LInux-LOader, le permite seleccionar, cada vez que inicie el sistema, la partición de la que desea arrancar (Linux, DOS, OS/2...). Para evitar posibles conflictos con el cargador de otros sistemas (p.ej el de OS/2) es mejor instalarlo en la misma partición de Linux (root partition), y marcar esta partición como la de arranque.

2) Un método opcional para arrancar Linux (y el más recomendado) es LOADLIN. Esta utilidad permite arrancar desde la línea de comando DOS. En este caso, instalar LILO no es necesario. Si lo incluye cuando el instalador se lo pregunte, setup le dejará los ficheros /root/loadlin15.zip y /root/loadlin15.txt. Lea la documentación adjunta cuando acabe la instalación general (se sale con 'q'):

```
less /root/loadlin15.txt
```

Suponiendo que ha permitido acceso a sus particiones MS-DOS, desde el directorio /dosc, en la instalación, copie el archivo comprimido y la imagen del núcleo de arranque (vmlinuz) a la parte MS-DOS.

```
cp /root/loadlin15.zip /dosc
cp /vmlinuz /dosc
```

Recuerde la configuración de sus particiones, para saber la que debe montar en el arranque (la que definió como tipo 82).   fdisk -l

Rearranque DOS (CTRL-ALT-DEL funciona). Descomprima con unzip, y siga la documentación adjunta.

3) Un tercer método es mediante un disquete de arranque, que se creará a lo largo de la instalación. Tenga prepa-

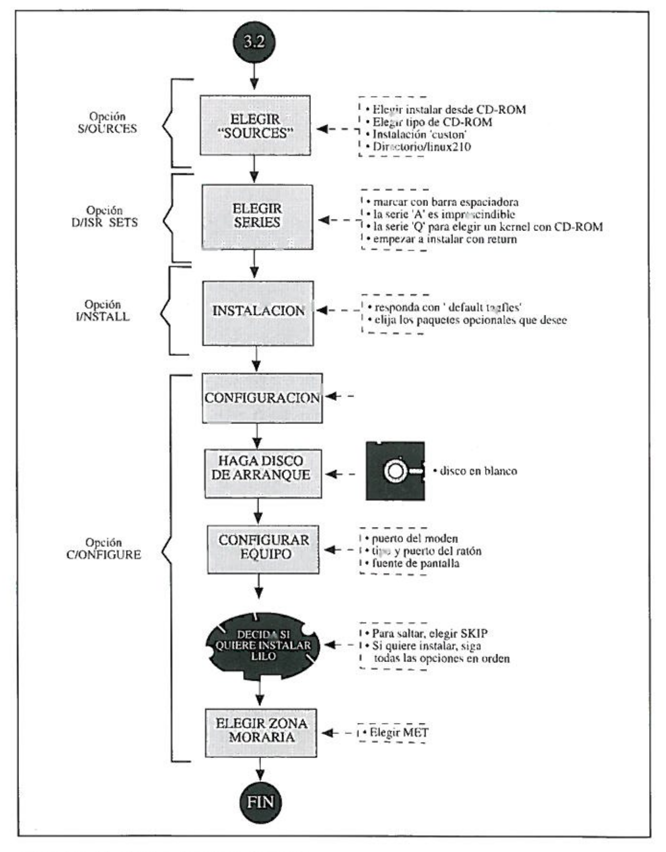


Figura 3. El programa Setup.

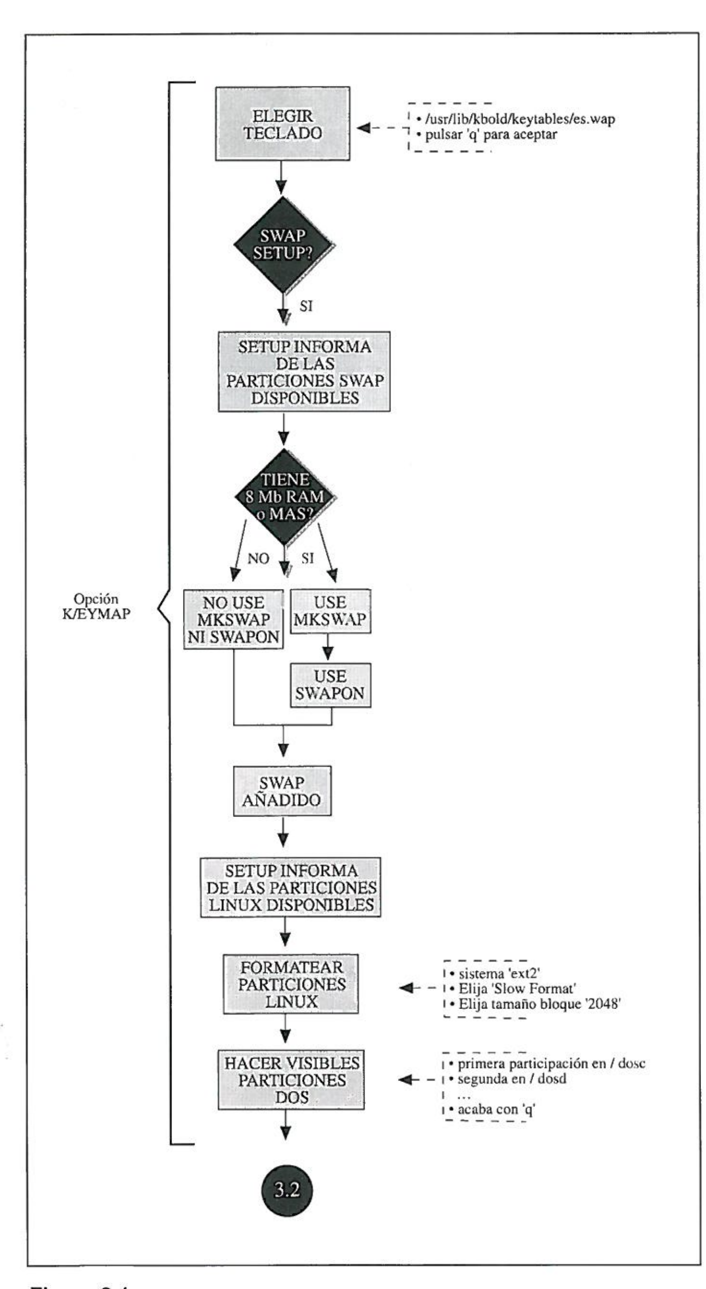


Figura 3.1.

rado un disco con formato, para cuando setup se lo pida. Este es el método mas sencillo, aunque, a la larga puede resultar algo incómodo.

AÑADIENDO MÁS PROGRAMAS

Si instala las X-Windows, observará que no puede arrancar las ventanas a la primera. Esto sucede porque, aunque se pueda averiguar la configuración de su tarjeta de vídeo, Linux no puede detectar las prestaciones de su monitor. Se adjunta un fichero ejemplo, XCONFIG.TXT, que supone una SVGA corriente a 640x480x256 colores y ratón Microsoft de 2 botones. Tómelo como un punto de partida y cópielo en: /etc/XF86Config. Para arrancar las X, teclee startx.

Para adaptar su hardware, lea la documentación (paquete xf_doc.tgz) en /usr/X11-/lib/X11/doc. También hay disponible información en el manual, como man XF86Config.

Además, aunque su teclado esté correctamente configurado en modo texto, las X-Windows necesitan su propio fichero para aceptar un teclado castellano.

Incluimos un ejemplo, XMODMAP.TXT, en el disquete. Cópielo como '/usr/X11/lib/X11/-xinit/.Xmodmap'.

Para usar su CD-ROM cuando todo esté instalado, y si no desea recompilar nada, compruebe la serie Q para buscar un kernel con el que arrancar a partir de ahora. Los kernels por defecto que se adjuntan en la serie A no incorporan los drivers para hardware específico, como lectores de CD. Se distribuyen dos versiones, la 1.18 (nombres terminados en 'o') y la 1.59. Lea la documentación correspondiente, si usa LILO o LOADLIN.

Cuando se desee añadir programas adicionales mas adelante, hay varios métodos:

1) Programas de las series no instaladas:

Utilice el 'setup'. Elija la serie que desee.

2) Programas sueltos (por ejemplo, del directorio \linuxutl)

Use directamente el comando 'tar xvfz <fichero_tgz>' desde el directorio en el que se desee descomprimir el archivo.

3) Programas en disquete.

3.1) Prepare un directorio vacío. Por ejemplo, "mkdir /dosa".

3.2) Monte el disco en dicho directorio:

mount -t msdos /dev/fd0 /dosa

3.3) Copie lo que desee a su disco duro

cp /dosa/* /root

3.4) Desmonte el disco ANTES de sacarlo de la disquetera.

cd

umount /dev/fd0

Cuando acabe la instalación, cree un usuario con adduser. Siga el ejemplo ADDUSER.TXT. Acostúmbrese a experimentar en el sistema haciendo login con este usuario y pase a root solo cuando sea imprescindible para hacer algo en concreto. Esto le protegerá del excesivo "poder" del superusuario. Es fácil despistarse y borrar información importante. Recuerde que (por el momento) no dispone de utilidades de recuperación de ficheros.

Compruebe que la instalación es correcta. Puede cambiar de sesión con Alt-F1 hasta Alt-F6. Entre con el login del usuario que haya creado. Observará que el prompt pasa de '#' a '$'. Salga con exit. Para apagar el sistema, puede usar ctrl-alt-del, o el comando "/sbin/shutdown -h now" como superusuario. Su sistema Unix ya está funcionando.

Aun así, es muy probable que se le planteen dudas sobre puntos concretos en el funcionamiento y la configuración. Siga los ejemplos del curso Unix, o acuda a la abundante bibliografía sobre Unix. En las series, la 'F' (FAQ: Preguntas frecuentes) y la 'I' (Info) pueden resolver sus dudas. Si tiene un módem, puede conectar a su BBS preferida. Hay muchos usuarios que ya han instalado Linux y pueden solucionar dudas específicas. Sobre todo, tenga paciencia. Unix es un sistema muy potente, pero al que es necesario acostumbrarse. ■

CORREO DEL LECTOR

En esta sección, los lectores de SOLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

P *Llevo un tiempo programando, pero en mi localidad hay poca gente dedicada a este tema, por lo que, aparte de comprar libros especializados, y de adquirir su estupenda revista, desearía contactar con otros programadores, para intercambiar datos y rutinas o, simplemente, charlar. Me han hablado de redes de mensajería y de programas, pero no se como contactar con ellas. Les agradecería que me informarán al respecto.*

Pedro Luis Ortega
(Cáceres)

R Aparte de Internet, con la cual es difícil de conectar "domésticamente", la otra gran red es Fidonet y es justo lo que le hace falta, pues tiene áreas públicas de mensajería de casi cualquier tema imaginable, como por ejemplo (relativas con la programación): Ensamblador, C, Pascal, Objetos, OS/2, C++, BASIC o Programación en general.

Todas estas áreas son a nivel nacional, pero existen otras tantas internacionales (se debe escribir en inglés), e incluso simplemente, algunas son locales.

Para contactar con Fidonet, consiga cualquier teléfono de una BBS (Bulletin Board System) conectada a Fidonet (la mayoría), conecte con ella mediante cualquier programa de comunicaciones y un módem (preferiblemente rápido) y consiga la lista de las BBS Fidonet de España, busque la más cercana a su domicilio (por cuestiones económicas, pues va a llamarla a menudo), y ya tiene acceso a la opinión de miles de programadores de todo el mundo. Por otro lado, en todas las BBS existen áreas similares pero de ficheros, en las que se puede encontrar prácticamente cualquier ejemplo, información o fuente sobre programación.

Aunque últimamente algunas BBS están solicitando colaboraciones para su mantenimiento, la inmensa mayoría son gratuitas, pues una de las reglas de oro de Fidonet es generalmente la filantropía.

P *Hola, el CD-ROM que incluían en el número 2, contenía unos ficheros con la extensión .DES que describían el contenido de cada directorio, pero se hecha de menos una utilidad que aglutine los ficheros con las descripciones, para así poder visualizar el contenido del CD fácilmente con solo un vistazo. ¿Podrían incluir algo en el siguiente?, un saludo.*

Ángel Cebreros
(Sevilla)

R Buena idea, pero no hace falta esperar al siguiente número, en el directorio CORREO del disco se encuentra LISTA.C, una utilidad sencilla que realiza justo lo pedido.

Y aquí tenemos el fichero BAT que genera la lista de fichero .DES para después llamar a la utilidad LISTA.EXE y redireccionar su salida al fichero DESCRIPT.TXT, que contendrá la descripción completa del CD.

CREALIST.BAT

```
dir f:\*.des /s /b > lista.txt
lista lista.txt >descript.txt
```

P *Exactamente ¿cuáles son los tipos de datos integrales (integral types) en C++?. Gracias de antemano.*

Andreas Metzger
(Madrid)

R Los tipos integrales son una parte de los tipos fundamentales, que son a su vez los que vienen definidos en el lenguaje propiamente dicho y lógicamente los más utilizados:

TIPOS FUNDAMENTALES:
Aritméticos:

Integrales o discretos: no se pueden fraccionar y todavía hay una subdivisión:
- Normales (char, short, int, long)
- Enumerados (enum)

Reales: son capaces de almacenar números fraccionarios de mucha precisión, también son llamados números de coma flotante:
- float, double, long double

Void (void type): significa "tipo nulo o vacío" y sus funciones son muchas y variables, es una especie de tipo de dato "comodín".

Los valores máximo y mínimo de cada uno de estos tipos (los aritméticos), están definidos en el fichero <limits.h> de cada compilador ya que son dependientes del procesador, e incluso, del compilador.

La definición exacta de los tipos reales, la encontraremos en el fichero <float.h>.











MINISTERIO DE DEFENSA
DIRECCION GENERAL DE LA GUARDIA CIVIL
MINISTERIO DE CULTURA
MINISTERIO DE ECONOMIA Y HACIENDA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
GENERALITAT DE CATALUNIA. DEP. SANITAT
SERVEI CATALA DE LA SALUT
BOLETIN OFICIAL DEL ESTADO

BANCO DE SANTANDER
INSTITUTO NCNAL. DE LA SEG. SOCIAL
MINISTERIO PARA LAS ADMINISTRACIONES PUBLICAS
FONDO DE GARANTIA DE DEPOSITOS
BANCO SANTANDER PUERTO RICO
BANCO ATLANTICO
PRICE WATERHOUSE
TOSHIBA

TELEFONICA
SANTANDER NATIONAL BANK
CORPORACION FINANCIERA HISPAMER
ANALISTAS FINANCIEROS INTERNACIONALES
SEGUROS OCASO
EUROSEGUROS BBV
SEGUROS ATHENA
FYCSA (ALCATEL)

CONSTRUCCIONES AERONAUTICAS (CASA)
PATENTES TALGO
SINTEL
AVIACO
AEROPUERTO DE CANARIAS
TRANSMEDITERRANEA
PUERTOS DEL ESTADO
GEC ALSTHOM
L OREAL

FASA RENAULT
RED DE CONCESIONARIOS RENAULT
REPSOL EXPLORACION
REFINERIA DE GIBRALTAR
PETROGAL
ONDA CERO RADIO
TELEMADRID









TELECINCO (GESTEVISION-TELECINCO)
UNIVERSIDAD AUTONOMA DE MADRID
UNIVERSIDAD CARLOS III DE MADRID
INSTITUTO DE EMPRESA
TELEFONICA SISTEMAS*
S.P. EDITORES*
OLIVETTI*

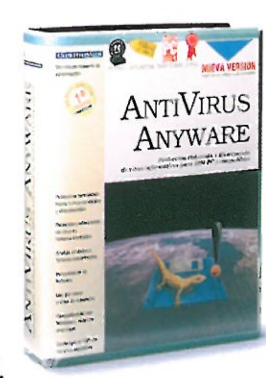
IBM*
SIEMENS NIXDORF*
DIGITAL*
BULL ESPAÑA*
FUJITSU*
INVESTRONICA*
SOFTWARE DE DIAGNOSTICO*

COMPUTER ASSOCIATES
G.P. INFORMATICA*
INFORMATICA EL CORTE INGLES*
ACTION*
GTI*
ALCATEL SISTEMAS DE INFORMACION*
INDAS

EL CORTE INGLES*
SOFTWARE DE ESPAÑA*
INFORMIX
ABBOTT CIENTIFICA
ALBILUX
ELIDA GIBBS
OXFORD UNIVERSITY PRESS

9 de cada 10 Ordenadores prefieren ANTIVIRUS ANYWARE.

El resultado está a la vista. O, ¿cree que tantas Empresas pueden estar equivocadas?. Tienen razones para no estarlo.



1. MAYOR PROTECCION.
Incorpora un Sistema de Protección con una ocupación mínima en RAM. Detecta el Virus antes de que pueda introducirse e impide el uso del fichero contaminado, avisándole a través de una ventana de alarma.

2. DETECCION INSTANTANEA.
De forma rápida y precisa, analiza cualquier unidad, directorio o fichero. Comprueba si su disco duro o disquetes contienen algún Virus. Chequea ficheros (incluso comprimidos), sector de arranque, tabla de particiones y unidades de red.

3. ELIMINACION DEFINITIVA.
Elimina los Virus allí donde se encuentren, sin dañar los ficheros.

4. SERVICIO DE ACTUALIZACION PERMANENTE.
Usted puede recibir las nuevas versiones cómodamente en su domicilio o capturarlas vía modem.

ANYWARE pone a su disposición el CLUB DE USUARIOS HELP VIRUS con una HOT LINE exclusiva para consultas, noticias, BBS, etc.

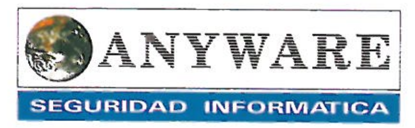









PREMIO PC MAGAZINE MEJOR UTILIDAD 1991 · RECOMENDACION PC WORLD MEJOR PRODUCTO ANTIVIRUS · RECOMENDACION PC MAGAZINE MEJOR PRODUCTO ANTIVIRUS · CATACOM 93 MEJOR ANTIVIRUS · PREMIO PC MAGAZINE MEJOR UTILIDAD 1993



Orense, 36 3º. 28020 MADRID. España.
Tel.: (91) 556 92 15. Fax.: (91) 556 14 04.

EMPRESAS E INSTITUCIONES CON LICENCIA CORPORATIVA DE ANTIVIRUS ANYWARE
(*) EMPRESAS DISTRIBUIDORAS DE ANTIVIRUS ANYWARE.